Probabilistic Post-Facto Detection of Man-in-the-Middle Attacks on Unauthenticated Diffie-Hellman

Matthew Johnston

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

2014

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Matthew Johnston
27th August 2014

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Matthew Johnston
27th August 2014

Acknowledgements

I would like to offer my thanks to my project supervisor Dr. Stephen Farrell for his help during the dissertation project. I would also like to thank my course director Dr. Stephen Barrett and Ms. Gillian Roddie of postgraduate support for their kind support and assistance in completing the Masters.

I would also like to offer huge thanks to my family and my cats for their love and patience during my studies. I couldn't have done this without them.

Another huge thank you to my housemates and friends; the residents of Trinity Hall apartment 82.02, 2012-2013, for reminding me to take a break and be human.

Lastly, to everyone I've met in Dublin who's listened to me explain this topic; I'm in equal parts grateful and sorry.

Matthew Johnston
August 2014

Matthew Johnston
MSc. Computer Science (Networks and Distributed Systems)
Probabilistic Post-Facto Detection of Man-in-the-Middle Attacks on Unauthenticated Diffie-Hellman
Supervisor: Dr Stephen Farrell
2014

This dissertation details a prototype protocol for exposing a man-in-the-middle attacker on Diffie-Hellman key exchanges at a scale where out of band verification of matching shared secrets is infeasible. This verification is accomplished by creating a hash value of the shared secrets as seen at each end point and comparing the outputs, where a mismatch in hash values indicates a potential man-in-the-middle. The exchange that each hash value must be able to be identified without divulging the actual identities of the participants.

Two systems for this are created and evaluated, each using a different method of identifying the exchanges in an online database and analysing the results of comparing hash values. One version uses random integers as part of the identifier, where the count of random value choice collision for matching hashes can be compared with the expected count based on the size of the random range. The second has servers create a UUID to record batches of DH exchanges and later compares this local log with the online version, which includes client versions of the exchange.

The random session version works well in the case where a number of participants provide information, offsetting the inaccuracy of the random numbers. While the number of exchanges under a particular range is up to an operator to decide, little modification is required to the underlying protocol.

The UUID version works with fewer participants, though requires more involvement of the operators and changes to the underlying protocol.

Contents

List of Figures

List of Tables

1 Introduction

The Diffie-Hellman key exchange protocol is a widely used method for communicating processes to create a shared secret key over an insecure channel, where it is computationally infeasible for an attacker to calculate the secret key with the transmitted data. In the unauthenticated and in some authenticated models of Diffie-Hellman exchanges, an attacker can interpose themselves in the key exchange and can then decrypt all transmissions as well as masquerade as each user. This kind of attack is called a "Man-in-the-Middle" attack.

These attacks can be difficult to detect, as a successful Man-in-the-Middle attacker will be transparent to each party and processes will see an encrypted tunnel existing between the parties, despite an attacker being able to manipulate the entire transaction. An opportunity to detect these kind of attacks at scale may exist in verifying the shared secrets are identical, as a Man-in-the-Middle is unable to guarantee that the shared secret at each end point is the same. The secret values themselves do not have to be compared to each other directly, only that the values are identical.

Comparing the shared secrets at scale is difficult, as the in-band communication is always susceptible to modification if a Man-in-the-Middle exists. Several protocols use an actual or virtualised out-of-band channel for users to verify the matching of shared secrets, though these are highly coupled to the medium used. There is a potential for a protocol to provide benefit if shared secrets can be verified to be identical in a way that an attacker can not easily influence. Especially if such a protocol can operate with a level of independence from the underlying protocol.

This dissertation presents a design for such a protocol, details the development of a prototype implementation and the evaluation thereof.

2 State of The Art

In reviewing the state of the art relating to post-facto detection of eavesdroppers in supposedly secure communication, several topics of research are useful to consider. Particular areas of interest include;

- The design, implementations and limitations of Diffie-Hellman
- Existing protocols designed to expose eavesdroppers in communications
- The reasons for creating such a protocol
- Technologies and techniques of use in designing a new protocol

2.1 Diffie-Hellman Key Exchange

Diffie-Hellman (also called Diffie-Hellman-Merkle, DH) is an method for the exchange of secret keys, proposed by Whitfield Diffie and Martin Hellman, with contributions from Ralph Merkle in 1976 [7, 8].

The Diffie-Hellman algorithm allows two or more parties that have no prior interaction to establish shared secret keys over an insecure channel, through exchanging non-secret information created by a function computed on secretly held random values. The generated secret key can then be used to symmetrically encrypt further communications between the parties.

The Diffie-Hellman key exchange protocol operates as follows; The two end-points, often referred to as Alice and Bob, agree on two non-secret, prime numbers, g and p, where p is a large integer value and g is a primitive root mod p. Alice and Bob independently choose large random integers, a and b, to use as their private key, which is kept secret and never transmitted over the channel.

With p and g, each party can compute a non-secret value as;

Alice: $A = g^a (\text{mod } p)$
Bob; $B = g^b (\text{mod } p)$

Alice and Bob can now share A and B with each other and use these to independently create the shared key as;

Alice computes;
$K = B^a (\text{mod } p) = (g^b)^a (\text{mod } p)$
Bob computes;
$K = A^b (\text{mod } p) = (g^a)^b (\text{mod } p)$

The shared key K can be used in a symmetric cipher to encrypt further messages exchanged between Alice and Bob [8, 30]. Figure 1 shows this exchange in a flowchart.

Prior to the development of Diffie-Hellman, a secure out-of-band channel was required to exchange key information, which required time and investment and was therefore inapplicable to a range of use cases. At the time of the development of Diffie-Hellman, a suggested use case was in automated teller machines (ATMs), which needed to be able to create secure keys for the transmission of bank data. As these ATMs could be installed in relatively remote locations, it was seen as important to be able to dynamically change keys without transmitting any secret information either from the ATM or the issuing bank. As technology advanced, the Diffie-Hellman exchange protocol was included in many Internet security protocol specifications, such as TLS, which are commonly used in e-commerce and securing private and personal information online.

The security of the Diffie-Hellman key exchange is achieved due to the complexity of calculating a discrete logarithm over a finite field, making it computationally and time intensive for an eavesdropper to calculate the secret

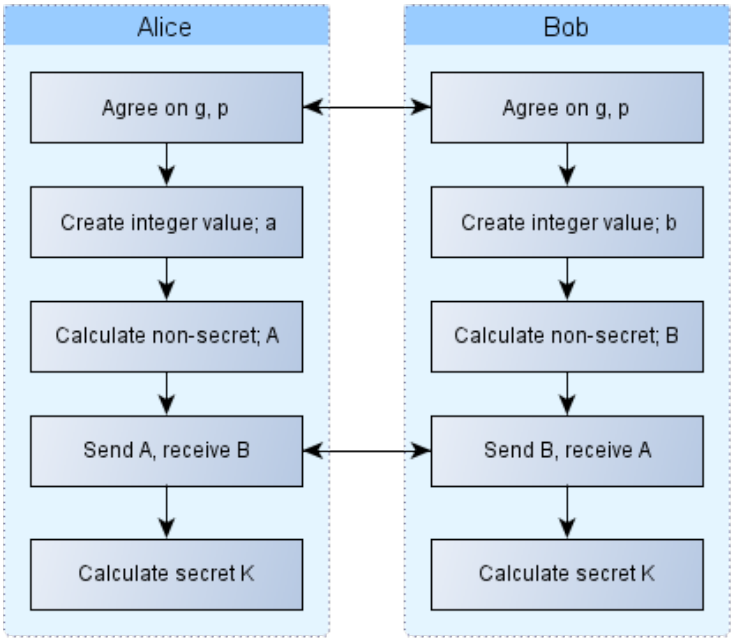

Figure 1: Graphical representation of the integer Diffie-Hellman exchange

key from non-secret information. Despite advances in computing power since the development of Diffie-Hellman, the discrete logarithm problem is still considered complex enough to provide security for modern usage [30].

2.2 Elliptic Curve Cryptography

The standard implementation of Diffie-Hellman, described in section 2.1, uses prime integers as part of the calculation for the shared secret and as is referred to as integer Diffie-Hellman. A potential change in coming years involves the use of Elliptic Curve Cryptography (ECC), an approach to public-key encryption based on the mathematical complexity of plotting points on an elliptic curve in addition to the discrete logarithm problem.

Alice and Bob can choose a publicly available elliptic curve pattern and a non-secret fixed curve point, F.

Alice can then compute respective private and public keys as;

Private Key: A_{prv} - Random integer

This integer is multiplied by the agreed fixed point, F, on the chosen elliptic curve to calculate another point, A_{public}, which is used as Alice's public key.

Public Key: $A_{pub} = (A_{prv} * F)$

Alice and Bob can then calculate the same secret key K using the following formula;

Alice computes;
$K = (A_{prv}) * (B_{pub}) = (A_{prv} * ((B_{prv})F))$
Bob computes;
$K = (B_{prv}) * (A_{pub}) = (B_{prv} * ((A_{prv})F))$

[16, 15]

The proposed benefit of Elliptic Curve Diffie-Hellman (ECDH) is in providing comparable security to integer Diffie-Hellman with smaller key sizes and significantly faster computation of secret keys [19]. This feature could be useful in providing efficient security in low power devices such as wireless sensor nodes and barebones style computers such as the Raspberry Pi [21].

ECC has not yet been widely adopted due to patents and intellectual property rights on some of the functions critical to the operation of the algorithm, however, when the patents expire ECC is considered likely to be adopted [17]. ECC is unlikely to fully replace integer Diffie-Hellman as many applications will require backwards compatibility, though new developments and packages may provide support for changing key agreement function.

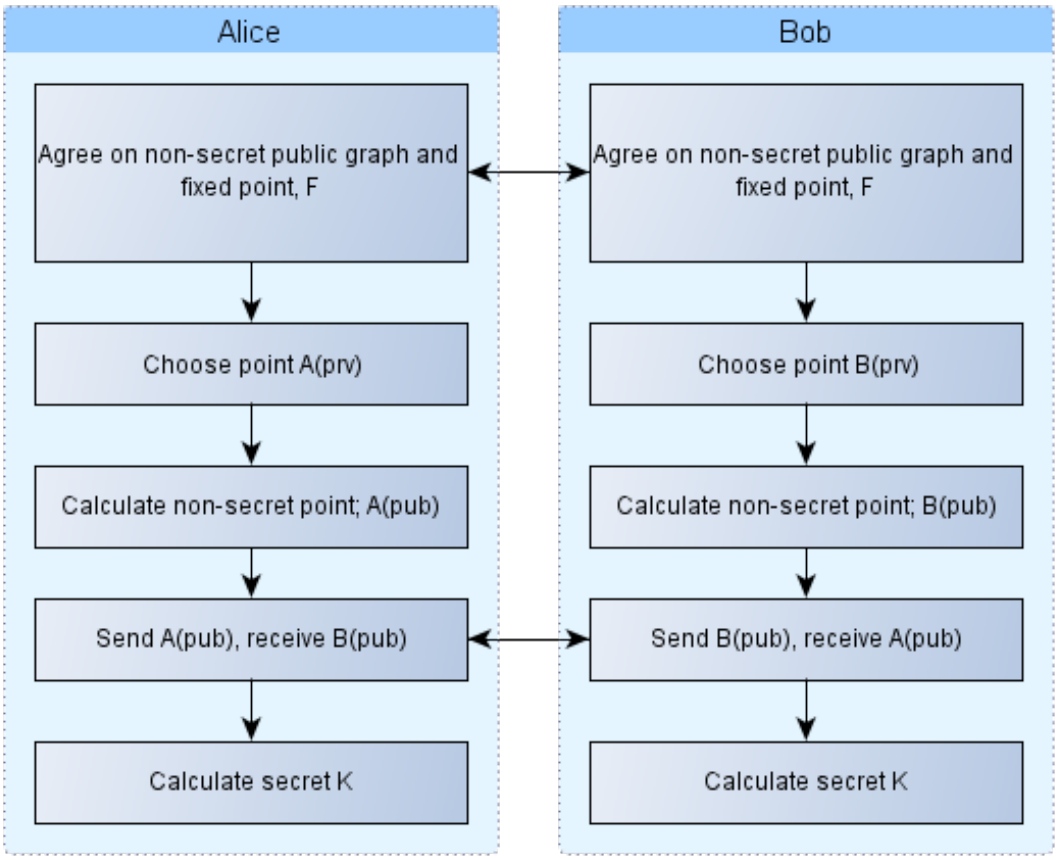

Figure 2: Graphical representation of ECC Diffie-Hellman

While the method for calculating the public and private keys is different, the outcome of the private key calculation still be used to create the artifact used to verify that the shared secrets are the same. This can be achieved by treating the key, or a subset of the key, as an integer data stream allowing the method by which it is calculated to be abstracted away.

2.3 Man-in-the-Middle Attacks on Diffie-Hellman

Petraschek, Hlvacs et al. [22] illustrate a thought experiment called the "Chess Grandmaster Problem", wherein a dishonest player could beat a grandmaster at postal chess by having a second grandmaster play a separate game as the opposite chess player to the first. The dishonest player would then show the first grandmaster's moves to the second as though it was the player had made the moves and the vice versa to the second grandmaster, creating the illusion the two participants at the end points are communicating directly to the other. This provides an analogue to a Man-in-the-Middle (MitM) attack on a communication between two parties.

During the secret exchange process, the Diffie-Hellman protocol is vulnerable to an attack where the attacker interposes themselves between the processes and participates in two separate key exchanges. The attacker can then set up an encrypted tunnel from each process to themselves, giving the end-points the impression of having a direct, secure connection to each other. If the attacker achieves this, they will be able to decrypt, modify, create and encrypt messages to and from each party and impersonate either user to the other.

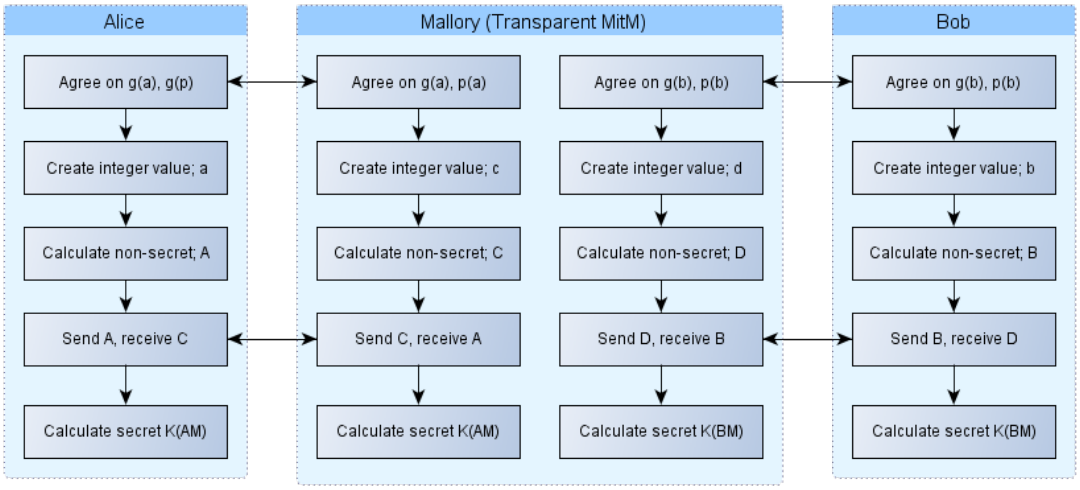

Figure 3: Graphical representation of an MitM in the Diffie-Hellman exchange

In the case of a Man-in-the-Middle attack, an attacker can set up shared secrets between itself and each party in the communication, but cannot ensure that the two shared secrets are the same. This affords a capability of comparing the shared secrets in a way the attacker cannot stop or modify, i.e. an out-of-band channel, in order to identify the presence of an eavesdropper. Several protocols exist that allow for such checks to take place through either physically or logically out-of-band channels, for example ZRTP, however at Internet-level traffic, such channels are unlikely to scale effectively.

The issue of Man-in-the-Middle attacks has been exacerbated by the discovery of widespread usage of such attacks by government agencies as revealed by Edward Snowden. News outlets [11] report that a previously secret operation, believed to be run by either the American NSA or British GCHQ, called FLYING PIG, which appears to show a Man-in-the-Middle attack operating against a Google router. The attack decrypts secret communications and copies meta-data about the communication for the intelligence agencies.

Various methods of mitigating or increasing the complexity of Man-in-the-Middle attacks at run-time exist, such as password authenticated key exchange (PAKE) and public key based operations, which allow processes to verify the identity of the process they are communicating with. However, Callegati et al. [6] showed that these methods have known issues, such as guessable passwords and self-signed certificates tricking end users into erroneously accepting the forged signatures. Certificate-based authentication is also vulnerable to corruption of the signing authority, as was the case in the Diginotar breach, where signed certificates were given to unauthorised users and exploited to gain access to privately owned routers [11].

2.4 Pervasive Monitoring

Pervasive monitoring is a term given to the widespread, non-targeted and often warrant-less monitoring of large volumes of data, typically carried out by major organisations, with or without the consent of the users, operators or owners of the system. Concerns have been raised by human rights groups, Internet service providers and the general public over the apparent lack of judicial oversight and underhanded tactics in the collection of this data and implications to freedom of speech and association. There has also been much investigation into how complicit, willingly or otherwise, that major Internet Service Providers were in the collection and sharing of such data. The scale of the surveillance and the range of the affected parties created concern and much discussion among security professionals, some stating that the issue of pervasive monitoring be treated as a direct attack on the Internet. [29].

Evidence for the government's use of pervasive monitoring of computer networks has been mounting in recent years. In 2006, the Electronic Frontier Foundation (EFF) received a testimony from Mark Klein, a former network engineer at AT&T [27, 3, 2], stating that a fibre-splitter had been installed by the NSA in a secure room, called "SG-3" or "cabinet 641a". This fibre-splitter created a copy of all of the Internet traffic coming through AT&T Internet services from its customers. Once this data had been collected, it could be sorted and content filtered according to user-defined rules and filters at the agent's discretion, implying that the original collection had not been targeted or limited as would have been under an official warrant. The EFF went on to say that from Klein's testimonial and evidence of other AT&T-hosted NSA deployments, that the equipment is far in excess of what would be necessary to monitor only overseas traffic, and that the NSA could access around half of AT&T's total traffic [2].

Revelations made by Edward Snowden to The Guardian [18] in 2013 state that British GCHQ has a secret project, codenamed Tempora, wherein GCHQ taps phone lines and Internet fibre-optic cable to collect, store and sort a vast amount of traffic. As of the public discovery in 2013, the Tempora operation is alleged to have been running for 18 months. Tempora is only a part of the far-reaching surveillance program revealed by Snowden, with other operations targeted at different social network platforms, service providers and hardware facilities such as cloud hosting and the exchanges between them. Some of the data may have been collected by the service providers for legitimate purposes, such as accounting or auditing, then acquired by the intelligence agencies for other purposes.

Other revelations made in the documents released by Edward Snowden included the identities of countries involved, the United States, the United Kingdom, Australia, New Zealand and Canada, to create the so-called "Five-Eyes" partnership. This partnership included co-operation on a massive scale in order to snoop on communications taking place over the transmission infrastructure in their territories [2]. The EFF quote figures from journalist Glenn Greenwald to illustrate the extent of the spying. These figures include; the NSA collecting 3 billion US telephone calls in 30 days, a single NSA "unit" collecting 97 billion emails and 124 billion phone calls, and a total of 13.5 billion pieces of data from India [27].

The large-scale tapping of fibre-optic cables is not a recent development, as emails published by Cryptome [9] suggest that secret wiretaps may have been installed on undersea cables as early as 2001. According to these emails, a nuclear submarine would have been used to plant the wiretap on the cable, indicating major support from military and government sectors over the years.

2.5 ZRTP

ZRTP is an extension for Real-time Transport Protocol, a method for the Internet streaming of Voice-over IP (VoIP), which provides functions for manually verifying that a Diffie-Hellman exchange took place without a Man-in-the-Middle [33] attack. This technique uses a cryptographic hash function over the shared secret, which the end users read and compare characters from the output strings over the VoIP connection, using it similarly to an out-of-band channel. If a Man-in-the-Middle is present, it is highly likely that the shared secrets will differ, affecting the hash output and alerting the end users [22]. The cryptographic hash must be short enough for human operators to compare digits, but long enough to make guessing impractical. ZRTP using base256 as the hash function results in an attacker having a 1 in 2^{16} chance of a correct guess. Much of the security afforded from ZRTP comes from the real-time nature of the communication, wherein it is very difficult for an attacker to interpose themselves quickly enough to avoid detection due to delays in authentication responses.

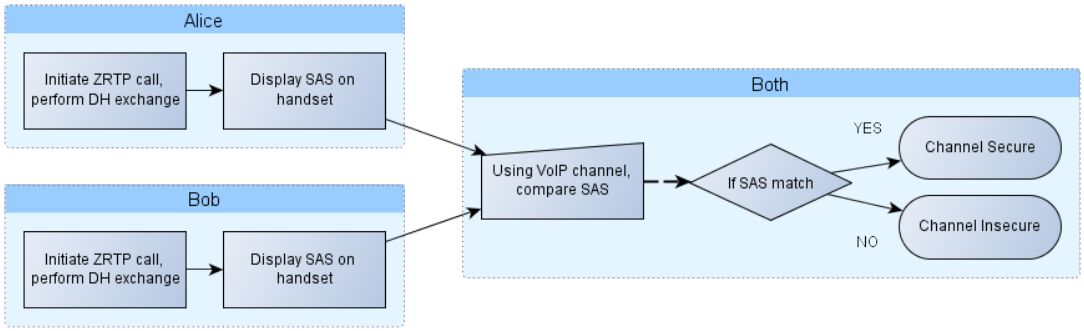

Figure 4: Graphical representation of ZRTP

ZRTP is limited by its focus on VoIP, where the voice channel is considered a separate channel to the connection initiation channel. In the case of text-based communication over some transport protocol, a separate out-of-band channel is unlikely to exist in a manner that will handle the scale of the proposed protocol.

Petraschek, Hlvacs et al. [22] identify a number of vulnerabilities that exist where the communication partners are not known to each other in advance. If the end-users do not recognise the voice of the intended recipient, an active attacker speaking in their own voice can more easily gain the trust if the end users and pass off as legitimate. This attack, and others like them, rely on tricking or confusing the end user, which is a concern in all security systems. A technical attack, which may need to be taken into account in the design of the proposed protocol is where an attacker forces the end user's handset to verify every time, annoying the user and complicating the process of differentiating legitimate users from illegitimate ones.

Despite these limitations, many of the concepts covered in the design and various implementations could be applied to the proposed protocol. The approach taken in forming the hash for the authentication string could be useful, especially if the string does not have to be human readable or memorisable.

2.6 Interlock Protocol

Rivest and Shamir [26] propose a protocol which creates a cryptographic authentication code based on the shared secret that was created during the initial exchange. This authentication code is sent in two parts between each party encrypted with the other's public key to be decrypted by the other party's private key. If an attacker is attempting to go without being detected, they will attempt to avoid modifying any transmissions. If each party waits to receive a half before sending the next half, the eavesdropper is unlikely to discover and/or modify the authentication code quickly enough to avoid detection. The authors state that an attacker should find it virtually impossible to create a message that can be decrypted by each end point in time to avoid detection based on timing differences in the exchange.

Another stated benefit of this approach is that it forces an eavesdropper to act non-transparently, which makes it more likely that they will be caught in the long run. The authors claim that forcing the Man-in-the-Middle to take an active role in modifying messages in stream would force them to reveal their presence due to errors in either timing or authentication codes under separate security methods. This kind of detection is especially useful in the case where an agency carrying out the pervasive monitoring of Internet traffic wishes to avoid detection, which is highly likely in most cases.

An issue with the interlock protocol is the necessity for $4n$ extra messages per n Diffie-Hellman exchanges, which at Internet-scale becomes a large burden. This is compounded by the necessity to wait for each half of the authentication code before sending the next, causing longer latencies in carrying out Diffie-Hellman exchanges and performance of the underlying protocol.

2.7 Cryptographic Hashes

One of the key aspects in computer security is verifying the integrity of the data in storage or transmission [14, 23]; that the data has not been modified by an unauthorised party or hardware or software fault. Cryptographic hashes are a popular way of achieving this and are used in Transport Layer Security (TLS) as message authentication codes, as digital signatures and for creating checksum values for downloaded files.

A cryptographic hash function can take an input with an arbitrary length and format to produce a fixed length output string. A secure hash function should create outputs that significantly differ from each other, even if inputs are very similar. For example even if two inputs only differ from each other by one bit, the hash outputs should be very different. It should also be computationally infeasible to craft a message that creates a specific hash, or to find two messages that produce an identical hash value.

Popular implementations of hash functions include MD5, SHA-1, SHA-2 and SHA-3 [24, 23]. There is currently dispute in security circles over the efficacy of MD5, as a potential crack has been found. While this crack is currently theoretical, experts believe that it is worth deprecating use of MD5 in favour of other available functions [31]. SHA-256 is currently the recommended standard for new developments.

Cryptographic hashes are widely used to "scramble" data in a particular way for storage, particularly of confidential user details such as passwords, so they are not directly human-readable. However if an attacker obtains a copy of database of the hashed data entries, they can perform an offline "rainbow table" attack. A rainbow table attack is where a table of cleartext guesses are ran through an identical hash function to identify hashed entries, as the hash outputs from two identical inputs will be the same. To mitigate this threat, a non-secret "salt" value can be stored alongside the hashed database entry [4]. When a user wants to compare a cleartext entry to the salted and hashed database entry, they combine the salt and the cleartext before running them through the hash function. In order to perform the same type of attack, an attacker has to perform the complete rainbow table attack with the salts for each entry, increasing the time it takes to calculate every value in the database.

2.8 Anomaly Detection

The detection of anomalies within a dataset is a large area of study in the field of mathematics, with technical implementations in on-the-fly and post-facto analysis of potentially very large datasets. Hodge et al. [13] produced a survey of various outlier detection methodologies based on Grubbs's [12] definition of an anomaly; "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs."

Outlier detection depends on the identification of patterns emerging in the data set on which it operates, based on statistical analysis of subsets of the data. Once these patterns have been established, any subset of the data can be compared against the understood pattern to identify those that deviate from the observed pattern. Detecting outliers in datasets can highlight areas in which unusual activity took place, which can then be put to further investigation.

Hodge [13] also lists a variety of areas in which anomaly detection is used, including; fraud detection, fault identification, image analysis and behaviour monitoring. Zhang et al. [32] conducted a survey of anomaly detection in wireless sensor networks, highlighting the role of machine learning and statistical modeling in autonomous detection of outlier data. The focus on wireless sensors is useful in this regard due to the necessity of low power consumption and efficiency of computation as a result of low battery lifetime, where the principles could be used to reduce the computational overhead against large datasets. Furthermore, the unreliable nature of the devices used can be seen as analogous to the unreliability inherent in Internet-scale data collection, submission and analysis, with similar solutions useful in each case.

In the protocol proposed in this paper, expected operation of the Diffie-Hellman protocol will be considered to produce baseline or "expected" readings, with Man-in-the-Middle attacks being treated as outliers [10]. Causes of outlier readings in the case of analysing cryptographic artifacts based on the Diffie-Hellman exchange include;

- Clients (intentionally or otherwise) submitting erroneous data to the database
- Pervasive MitM attack causing differences in collision rates
- Network failure
- Hardware or software failure

The rates at which these are encountered are highly dependent on the use case of the proposed protocol and the protocol that makes use of the Diffie-Hellman exchange, so no one set of readings can be treated as a concrete baseline across all implementations. Further details on causes of variations are listed in section 3.1.2. There are also likely to be fluctuations in the patterns due to the bursty nature of communication on the Internet and various hardware or software faults causing ongoing differences, so human understanding is required to make a decision based on the findings.

3 Experiment Design

The experimentation is be split into two major areas;

- Developing a prototype post-facto detection protocol
- Simulating probabilistic detection of an MitM at scale

The proposed protocol has been given the title "MAYBE", a backronym for "MitM- Are You Being Eaten?". Developing an instance of the MAYBE protocol will demonstrate the viability of capturing useful information, anonymising that data to protect privacy and using that data to draw conclusions about the state of the network.

Simulating the probabilistic detection will be used to judge if pattern analysis the small scale of the experiment can be scaled to implementation at larger deployments.

The MAYBE protocol has a number of key functional areas needed to operate effectively. These include;

- Capturing information from the DH exchange to create a "MAYBE tuple"
  - A method of identifying data from the same exchange without divulging participant identities. Hereafter referred to as a session ID
  - An artifact of the DH exchange that can be used to verify the shared secrets are the same. Hereafter referred to as a witness value
- Storage of data and interpretation of results
  - Converting information in the database into an heuristic diagnosis of the existence of an MitM
- Addressing vulnerabilities in the implementation of the MAYBE service

A high-level overview of the construction of a MAYBE tuple can be seen in figure 5.

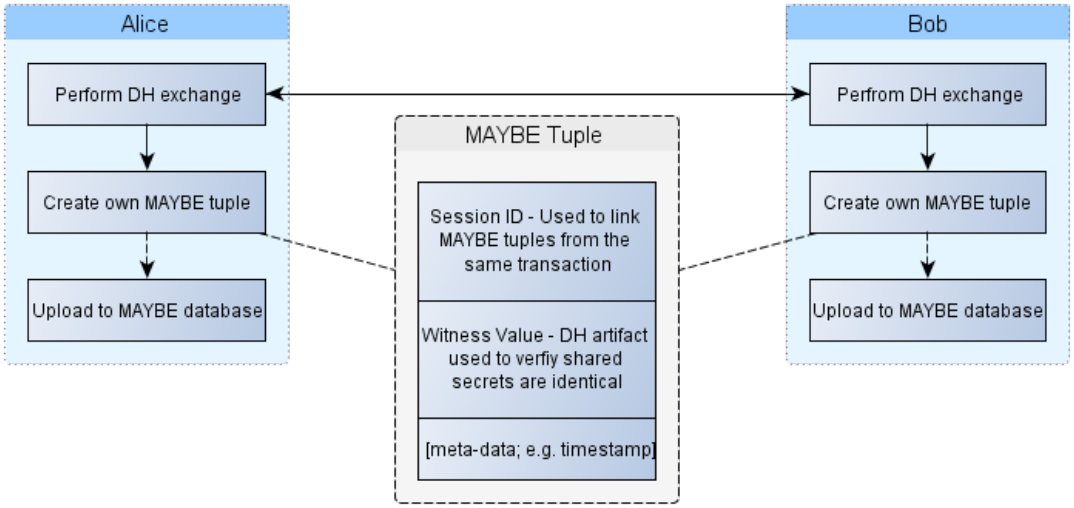

Figure 5: Structure and creation of a MAYBE tuple

3.1 Requirement Analysis

3.1.1 Creating Session IDs

As the database will contain a potentially very large set of information that could be privacy-sensitive, security concerns regarding this data must be considered in the design for the protocol. In a naive implementation, this information could be used to infer the identities, volume of communications and timing of communications of participants.

To protect the users providing information to the database, cryptographic hashes could be used to create one-way hash of identifying data in a way that an attacker is unable to easily infer data from, but act as a database search key for users supplying the information to locate their entries in the database.

Choosing the inputs to create this session ID is non-trivial, as any in-band communication is susceptible to manipulation by a MitM, yet both end-points should ideally calculate the same value. Several possibilities to create session identifiers to use as a session ID were considered, illustrated below;

- *IP/Port No.* Using some combination of the Internet Protocol (IP) address and port number of the communicating parties as a session ID, potentially with an agreed salt or timestamp to differentiate multiple DH key exchanges between two parties. Due to the common use of network address translation (NAT), it is possible that either or both parties will see the NAT-box IP address and port instead of the endpoint IP address and port, resulting in mismatched Session IDs. IP addresses remain a valid choice if it is known the parties communicating are not using NAT.

- *DNS Name* Potentially more resilient than using IP addresses, the Domain Name System (DNS) name for the intended end-point servers or services could be used similar to IP addresses detailed above. However, this relies on the security of the DNS and its resilience to various attacks to provide identical session IDs. However, if the parties are aware of the limitations of using DNS entries, then using DNS names could be a useful solution for small-scale implementations of the MAYBE protocol.

- *Timestamp* A timestamp would be useful as part of the generation of a session ID, as it could be used to differentiate repeated DH key exchanges between two endpoints. The granularity of these timestamps need to be set coarse enough to deal with time synchronisation in distributed systems, yet fine enough that the timestamp values are not easily guessable. The timestamp must not be usable to infer the identities of the endpoints by comparing the stored values with DNS lookup information.

- *Assigned UUID* The operator of the MAYBE server could provide participants with a set of difficult to guess Universally Unique Identifiers (UUIDs) for use as part of their session ID, which could be concatenated together to create a unique session ID over an TLS connection. This creates additional computation and communication overhead on all parties, and the endpoints must trust the MAYBE operator, who would have information on the parties exchanging encryption keys. This could be considered in organisation-scale use cases, where a network administrator with some level of escrow is already accepted.

- *Application Layer Protocol* Some identifier from the protocol that the Diffie-Hellman exchange process is protecting could provide a part identifier, for example SSH or IMAP could have agreed ID. These IDs cannot be used alone, as many protocols are reused between two parties.

- *Random Integer Range* Having each endpoint choose a random number within an agreed range will not result in each party choosing the same number every time, however, a MitM would be unable to influence the decision. This method could scale better than the other options as there is less processing, management and communication overhead. However, many of the entries in a database in this system will be mismatched, so the chance of detecting a specific instance of an MitM is lowered under this system alone.

Ideally, a combination of the above methods could be used and the combined information converted into a single hashed string. Potentially a combination of timestamp, DNS names of participants and a short random could be used together to make a session ID that has little chance of re-occurring, especially in a single investigation period. The short integer is used to configure an "expected", or ideal level of hash collisions of session IDs in the database.

For the purposes of this prototype protocol, random integer ranges will be used to create the session ID. This choice was made due to the constraints of this project and to provide data which can be used to extrapolate the effectiveness of the hash collision comparisons.

3.1.2 Random Session ID MAYBE

Throughout the document, the following terms will be used to describe the results of comparing database entries that form tuple pairs that represent each side of the Diffie-Hellman key exchange. For each of the tuples in the MAYBE database, there could be four events;

True Positive	MitM present & session IDs mismatch for same witness value
True Negative	MitM NOT present & session IDs match for same witness value
False Positive	MitM NOT present & session IDs mismatch for same witness value
False Negative	MitM present & session IDs match for same witness value

Table 1: Possible tuple session ID comparison results for tuple pairs in the random session version of MAYBE

Even in intended operation of the MAYBE protocol with no MitM present and all exchanges uploading both sides of an exchange, most entries in the database will be a false positive, as defined in table 1. This is expected and intentional, as changes in the ratio of true negatives to false positives are key to the probabilistic detection of a pervasive MitM.

A false negative in this scenario is considered unlikely unless the attacker deposits multiple values in the database, which can be identified as a separate attack. While such an attack is not directly indicative of a Man-in-the-Middle attacker, it should arouse the suspicions of the database operator and the contributing parties.

Methods of detecting and preventing this kind of attack could include;

- Submitting to the MAYBE database requires a login
- More than the expected number of entries for a single witness value raises an alert
- Unusual differences in timestamp values
- Unusually large amount of deposits from a single IP address or range

Other methods to protect databases available on the Internet should also be considered, such as authentication (depending on use case) and Denial of Service (DoS) protection methods.

In the case where a random session version of MAYBE is used, the randomness in the search key means a false positive in a tuple pairing match is not only possible, but relatively likely. Therefore, in order to draw conclusions on the presence or absence of a man-in-the-middle, as statistical analysis must be performed on a subset of the data, in order to ascertain the ratio of true and false positives to true and false negatives. With a two party implementation of Diffie-Hellman and a range of integers, I, the expected number of false positives in a set of tuples representing T Diffie-Hellman key exchanges is calculated as;

$$\text{Expected No. Positives} = \frac{T}{I}$$

This formula assumes all tuples in the data set have used the same I and that both parties have deposited their respective witness values. Values for I and T will vary greatly depending on the deployment and scope of the MAYBE deployment. Discovering the most effective balance of I to varying levels of T will be a useful area of inquiry. The value of the expected number of positives will also be referred to as the expected collision value (eCV).

If the difference between the expected and actual number of positives found from the data set varies by more than a certain tolerance, an alert should be raised. Possible results of the comparison operation can be seen in table 2.

Condition	Heuristic
Expected >Actual	Possible database manipulation
Expected ≈ Actual	"Expected" operation
Expected <Actual	Possible pervasive MitM

Table 2: Database search results and heuristic rules for the random session version of MAYBE

As the actual collision value will vary even when the number of Diffie-Hellman exchanges, random range and upload success rate are the same, the threshold at which heuristic decisions are made must be set accordingly. These threshold values must be set at points where intended operation is unlikely to trigger an erroneous MitM detection.

3.1.3 Session ID Alternative - CallMe

If participants are willing to relinquish some of the anonymity in using session IDs as in section 3.1.1, additional information could be captured in the MAYBE tuples to improve the quality of the detection methods. A variant using a string identifier, termed a "CallMe", as the session ID for a group of transactions to create an identifiable subset within the data set. This CallMe can be changed at intervals, and a range of identical CallMes used to identify a subset of Diffie-Hellman exchanges, taking place under certain similar conditions, such as the same server or the same domain.

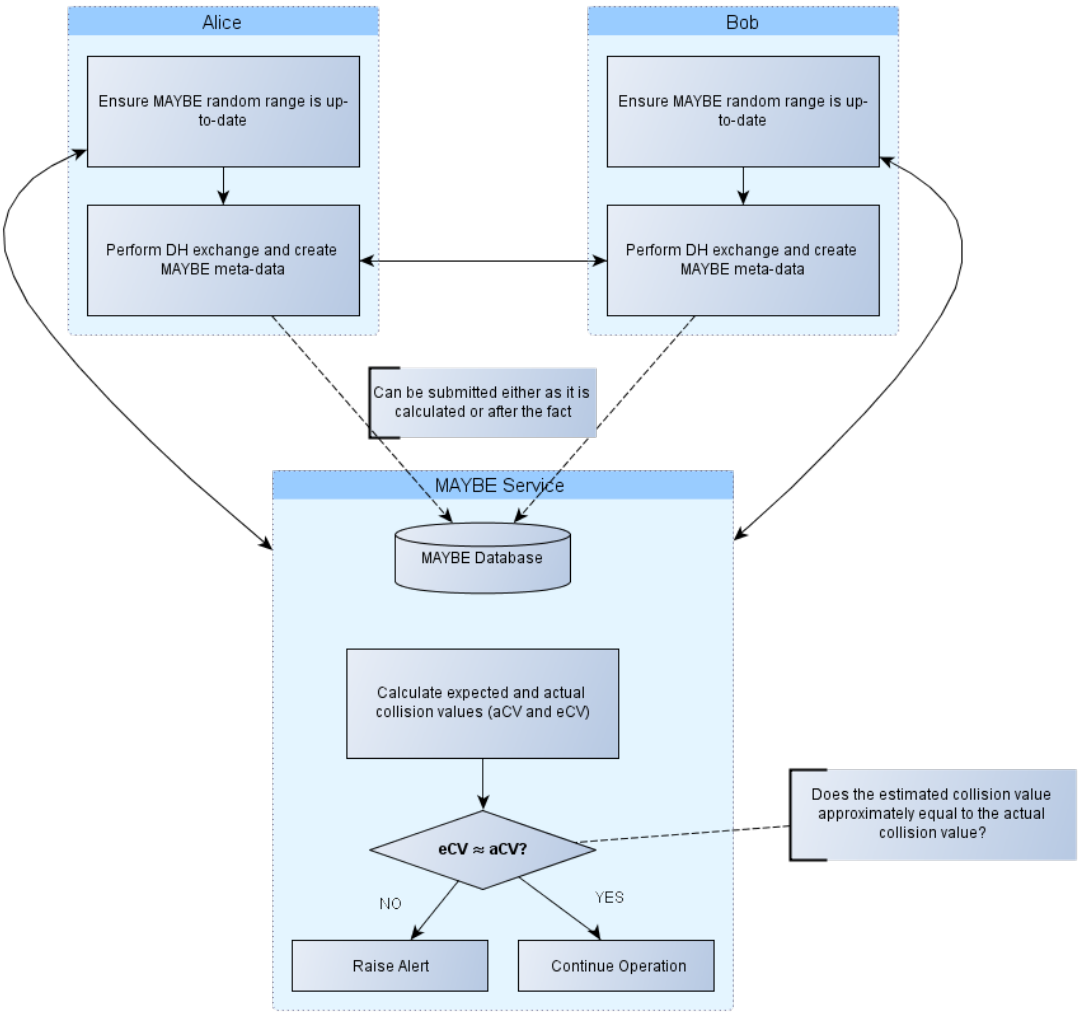

Figure 6: Graphical representation of client DH exchanges and MAYBE deposits with heuristic detection in the CallMe version

CallMes can be seen as analogous to using UUIDs as described in section 3.1.1, though with some level of delegation available in the choice and construction of the string, depending on the implementation of the MAYBE system. It should also be noted that this version requires changes to the protocol between the participants carrying out the DH exchange, whereas the random session version does not.

Along with the CallMe value, the witness value and associated meta-data create the MAYBE tuples in the public database similarly to the random session version MAYBE database. Storing this information anonymises some of the privacy-sensitive data, yet allows more granular analysis of the patterns in deposits. These analyses could be used to identify pervasive man-in-the-middle attacks or if an attacker is aware of the existence of the protocol, discourage the opportunistic interception of transmissions.

The CallMe version is hypothesised to work in a more distributed manner than random session, where many separate MAYBE servers can serve a subset of the total transmissions in a network. Depending on the underlying protocol that Diffie-Hellman exchange is being used to secure, different ratios of deposits from server and client could be expected and these differences could be allowed for in having different MAYBE servers for different protocols. The use of CallMes to differentiate groups also allows the possibility to have different protocols committing to the same MAYBE database, with the CallMes differentiating between them.

The most important part of the protocol is that the server maintains a log of the Diffie-Hellman exchanges, in particular the witness values, associated with each CallMe that they have used. This log contains the local knowledge of the exchanges that took place, which can then be compared to the MAYBE database, and if available, the respective client tuples. While there are other detection metrics available in this version, as described in section 5.2.1, this comparison of local logs to the database is where the strength of using CallMes is demonstrated.

The source of the CallMe strings could vary depending on the use case of the MAYBE deployment. A MAYBE operator may choose to assign particular CallMes to participating servers in order to manage their usage centrally, allowing for easier detection of fraudulent CallMe data.

However, centralised management does not scale well to Internet-level deployment, though may still be applicable at the scale of organisations and businesses. An alternative method is to allow server operators to assign and manage their own CallMe strings, and treat the MAYBE database as simply an external repository. This scheme delegates more authority and responsibility to the participating servers to keep accurate logs and change CallMes appropriately, though the delegation improves scalability.

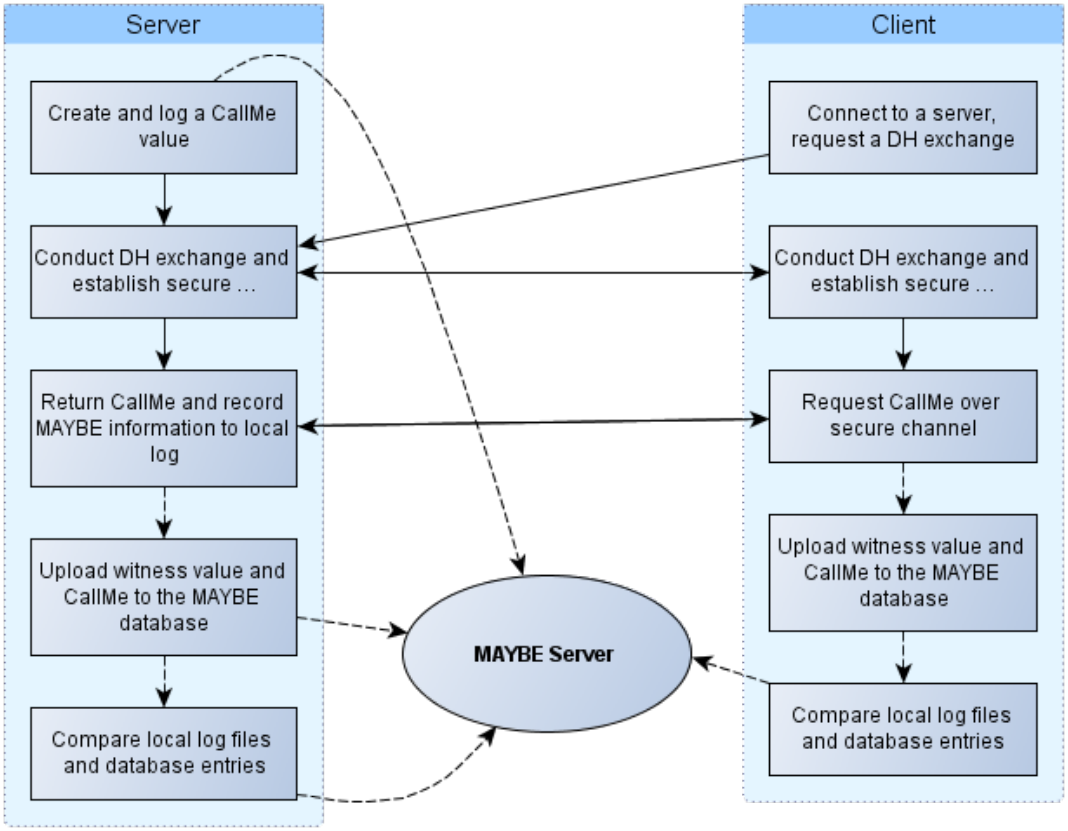

Figure 7: Flow chart describing the the CallMe version, showing the processing and interactions between the MAYBE server, participating DH exchange providers and clients

Using CallMes allows the MAYBE protocol to be used for different protocols with different characteristics that would affect deposit rate, such as reliability of connection, hardware and software heterogeneity and scale of deployment. This would lead to variances in the tuple matching rate even without the interference of a pervasive MitM.

Using the CallMes to identify sessions rather than the randomised integers in the design proposed in 3.1.1 places different requirements on the users of the MAYBE service, where the MAYBE participants providing DH exchanges keep a count of how many DH exchanges occur under a given CallMe. This count, n, can then be used as a base value upon which to perform various calculations on collected datasets as table 3 illustrates.

DB count: CallMes	**Heuristic**
$\approx$*(2n)*	Expected operation with fluctuation tolerance
n	Only one party is contributing witness values
<(n)	The database is being manipulated
>(2n)	More witness values than expected, possible naive MitM

Table 3: Heuristic diagnoses based on possible results from comparing the total number of DH exchanges under a given CallMe, n, and the number of records in the MAYBE database for that CallMe.

These heuristics can be converted into SQL queries in a similar fashion to those in the random session ID versions as described in section **??**. As there is more meta-data stored regarding each Diffie-Hellman exchange, more metrics are available to detect a man-in-the-middle attacker across a data set, as well as making it more difficult for any attacker to deposit fake values whilst avoiding detection.

3.1.4 Hypothesised Threat Model

As discussed in section 2.4 on Pervasive Monitoring, the attack the proposed protocol must address is the non-targeted, dragnet-style Man-in-the-Middle attacks against large volumes of transactions. The agents responsible for this kind of attack are likely to be well funded and supported, intelligent, motivated and aware of the existence and operation of both the proposed MAYBE protocol and the underlying protocol. Despite these advantages, in most cases the attacker will also need to act transparently for periods of time and adapt to avoid detection from a variety of actors, such as end-users, system administrators, software developers and service providers. Most organisations responsible for carrying out this kind of dragnet surveillance have a strong motivation to remain undetected, as discovery has lead to public outrage and political pressure in the case of the Snowden revelations and criminal prosecution in others. Compounding this, if their target becomes aware that they are under surveillance, they will likely change their behaviour or take technical, operational or legal steps to avoid further interception of their communications.

As the proposed attacker is likely to be aware of the existence, operation and even parameters of the MAYBE protocol, it is highly likely that they will take steps to avoid detection in this manner. The agent or organisation could modify their attacks in response to the parameters governing the operation of the protocol, for example, choosing opportunistic attacks or attacking fewer total Diffie-Hellman exchanges to lower their presence on the rate of session ID matches. This could be considered beneficial, as the attacker must make guesses as to the "safe" amount of traffic they can interfere with to avoid detection instead of capturing everything and as such, a subset of the exchanges taking place would take place without interference. However, some methods should be put in place to further complicate this process for an a attacker and increase the likelihood of detection.

A relatively simple method could involve changing the values of I and T across all parties participating in the MAYBE database at semi-random intervals, though this has scaling issues in updating clients providing information to the database. Regardless, forcing an attacker to take an active role in attempting to avoid detection could be considered a minor victory, as it both increases the complexity of the attack and makes an attacker more likely to be caught by other measures.

An opportunity exists for operating a protocol with a low detection probability (less than 1%) as a "canary", a process that can be executed in a batch operation to detect possible Man-in-the-Middle attacks on a very large amounts of data. "Canary", as used here, in reference to canaries in coal mines, where their sensitivity to the harmful conditions served as an early warning of a dangerous situation. As above, an attacker aware of this kind of collection of data must take care that they do not affect enough of the participating parties to be detected. However, without information about the volume of transactions to be monitored or the expected percentages, an attacker has to make a complicated series of guesses on a number of different variables affecting the ratio of expected to actual collision values. This kind of operation could also benefit from server operators with different instances of MAYBE using the same random ranges to compare data sets offline without divulging information at run-time, such as the timings of transactions.

3.1.5 Vulnerable Meta-data

Meta-data created during the Diffie-Hellman exchange must be protected, as this information could be of use to an organisation carrying out pervasive monitoring of communications. This meta-data could constitute a breach of a user's privacy even if the actual data exchanged remains secure. Pervasive monitoring agents target this meta- data, such as the number, direction, pattern and duration of connections to infer information that users would like to keep private [28].

The format and content of the tuples in the MAYBE database should also take into consideration the sensitive nature of the transmissions they are used to protect and be designed appropriately to avoid revealing this information. Therefore, information used for each field in the database should not offer a third party any advantages in learning about the nature of the transactions. Section 3.1.1 details how the session ID is created and section 3.1.6 illustrates the details of the witness value; the artifact of the secret exchange that can be publicly posted and compared.

The meta-data fields associated with the session ID and witness value must also be designed to not divulge privacy-sensitive information. The timestamp associated with the MAYBE tuple must be set at a granularity that complicates the process of comparing timestamps from the database with information such as the timing information on DNS name lookups to correlate tuples with identifying information.

3.1.6 Witness Value

The witness value that will be uploaded to the publicly viewable MAYBE database is an artifact of the shared secret that is established during the Diffie-Hellman exchange. As an MitM is unable to guarantee that the endpoints affected will generate the same shared secret, the difference in the artifact produced will reveal them or force them to act in a non-transparent manner to avoid detection [33]. This principle has been used successfully in other protocols, such as ZRTP as described in section 2.5. A secure hash algorithm, e.g. SHA-256, will be used to create a value that can be uploaded to a public database without compromising the security of the key.

3.1.7 Bulk Upload

The design of the MAYBE protocol allows for tuples to be uploaded after the Diffie-Hellman exchanges have taken place, instead of during the key exchange process itself. Clients or system administrators responsible for those clients may choose to delay uploading to the database for a number of reasons, including;

- Delaying upload of recently used keys
- Avoiding busy times in their local network
- Efficiencies in authentication methods for uploads
- Various other management or administration concerns

This necessitates a mechanism for uploading files containing records of DH exchanges, each record with the required information for the MAYBE tuple. Common technologies used to implement bulk upload include XML files and CSV files. Both of these require a pre-defined format for capturing and storing relevant information for later upload even if the transaction has been finished and the key deprecated.

Enabling bulk upload creates vulnerabilities that must be addressed, allowing an attacker to affect the database in various ways, such as;

- Bulk upload of falsified information
- Replay attacks of previously observed CSV files
- Modifying legitimate CSV files in transmission
- Impersonating a MAYBE server to receive CSV files from legitimate clients
- False positives caused by difference in bulk upload times of participants

Some of these concerns could be alleviated by having user authentication and data integrity checks on bulk uploads. Using public key cryptographic techniques for mutual authentication and message authentication codes could increase the protocol's resistance to these kind of attacks by ensuring the integrity of the MAYBE information and the identity of the submitter. However, this requires the participants provide some identifying data, even if only visible to the MAYBE server operator, which may be undesirable in some use cases.

3.2 Security Considerations

An important trade-off in the design and implementation of MAYBE is the need and desire for privacy versus identification of parties submitting to the database for auditing and authentication. Therefore, careful consideration must be taken over vulnerabilities that may be created in developing and deploying the MAYBE protocol and how best to address these.

The security considerations addressed in this section cover both protecting the MAYBE database against attackers and protecting confidentiality of the participants and their respective Diffie-Hellman exchanges.

3.2.1 Database Security - Depositing Witness Values

A basic attack against a naive implementation of a protocol similar in intent to MAYBE, would be for an MitM attacker to deposit session IDs and witness values for the attacked endpoints. This is addressed using session ID with a long enough random range that an attacker would need to deposit a large number of guesses in order to match the real session ID. Spamming the database in this way is not directly indicative of a MitM attack, but will arouse a great deal of suspicion, something an agency engaged in pervasive monitoring will wish to avoid.

Methods for detecting these kind of attacks exist in technical implementations for protecting Internet accessible databases as discussed in section 3.1.2. This kind of attack could also be detected if a single witness value has more than the expected number of entries with different session IDs, especially if the timestamps are close together.

3.2.2 Database Security - Random Database Spam

Similar to depositing witness values with different session IDs, spamming the database even with completely fabricated data will create an overhead on the database, as a large number of tuples could be inserted in a short period of time. Due to the suspicious nature of an attack like this, an alert should be raised to the MAYBE database operator. It may also be beneficial to have an automated defence, preventing overload while retaining log information that could be used in an investigation. Many server technologies provide some level of functionality for detecting and dealing with spam.

However, the defences need to be balanced against an attacker firing many requests and stopping any legitimate requests from being added to the database. Again, the techniques discussed in section 3.1.2 will be useful in mitigating these kind of attacks. Key properties in choosing the specific technique will be focused on detecting and discarding fraudulent data while allowing legitimate users to continue as intended.

If the users are willing to surrender some anonymity to increase the accuracy of the data, different methods could be implemented to authenticate users and make differentiation of legitimate submissions easier. Even if this data is not stored, complicating the process of mass uploading tuples could dissuade opportunistic attacks of this nature.

3.2.3 Database Security - Prevent Database Read/Write

If an attacker is able to prevent MAYBE protocol users from writing their data to the database, they may be able to avoid detection based on session ID/witness value comparisons. This is a much more active attack but could be used by various entities in parallel with MitM attacks, especially in the proposed case where the agency responsible for the pervasive MitM attack is well funded and supported. Attacking the MAYBE database, or the path from the users to the database is a completely different kind of attack with various additional difficulties, though not insurmountable to a skilled and dedicated attacker.

Evidence of this kind of attacks could be seen in the timestamps of the session tuples, where a sudden gap in otherwise regular updates could be seen. This is dependent on the usage of the underlying protocol that is being protected by the Diffie-Hellman exchange, as the bursty nature of many communications may make this kind of prediction difficult or impossible.

If clients keep records of the data that they have uploaded, or at least believe to have uploaded, querying the database for that specific tuple would show if the information had been uploaded and saved successfully. If data is removed from the data for legitimate reasons, such as database size management, querying for a deleted tuple should not cause a false alert.

3.2.4 Database Operators - False Alarms

A common attack against any security protocol is to repeatedly cause false alarms in the hope that the operator will disable or weaken parts of the security service out of sheer frustration. In this scenario, if an attacker can mask their identity and intention could repeatedly cause minor “nuisance” alerts, they could reduce trust in facets of the MAYBE system.

Examples of this kind of annoyance attack include;

- Annoying a MAYBE operator into relaxing tolerances on detection methods
- Annoying clients into ignoring or withdrawing from the MAYBE system
- Causing dynamic variables such as tolerance values to creep up or down

Reducing the total volume of alerts is a key element of mitigating these kind of attacks, as well as a keen understanding of the threat model the protocol will be encountering. As in section 3.2.2, an automated method of catching repeated entries into the database is needed, as well as alerting the system operators to a possible side-channel attack.

Participants and MAYBE operators should be informed as to the nature and capabilities of the attackers and the tactics that they may use in order to affect the MAYBE service. This is especially important as the adversary will likely be well aware of the existence of the MAYBE protocol and seek to find ways of reducing its effectiveness.

3.3 Design Choices

3.3.1 MAYBE Database Implementation

For the purposes of this experiment, the database will be implemented as a monolithic entity, hosted separately to the parties taking place in the Diffie-Hellman exchanges. This choice is made to reduce the overhead of managing a distributed solution and focus on the key feature of probabilistic detection of pervasive MitM attacks based on available data. This implementation is less likely in an Internet-scale deployment, where replication, security, availability and performance are key concerns, but may be seen at organisation or corporate level.

Further research could consider how distributed system principles could reduce the load on individual database operators, as well as make it more difficult to coerce the operators of the database.

In a real-world implementation, this monolithic database could be operated by an existing organisation that is trusted by many parties, such as the Electronic Frontier Foundation. The choice of host would be important, as the reputation of the operator will affect the public's trust in the results from the MAYBE calculations. The database operator must also be trusted to not add falsified data or remove legitimate data in order to affect the calculations.

The database used in the experiment could also represent a MAYBE implementation where each MAYBE database acts separately from other MAYBE databases, where different Diffie-Hellman transactions go through different databases.

3.3.2 Source and Range of Random Integers

The source and the range of the random integers from which the participants select a session identifier is highly dependent on the use case of the MAYBE protocol and the nature of the participating communications. The parameters of the random range as well as the frequency of changing this is a decision left to the MAYBE database operator, who should consider the operation environment of the participating clients. The factors to take into consideration include;

- Volume of Diffie-Hellman exchanges
- Risk factor of a pervasive MitM
- Ability to update participants (uptime, dissemination of information)
- Existence and operation of collaborating MAYBE servers

The random range could be set by the database operator or set dynamically based on the number of the entries coming into the database, with wider ranges corresponding to a higher volume of Diffie-Hellman exchanges.

Changing the range and volume of tuples included in the calculation could be beneficial to prevent the attacker from being able to ascertain a comprehensive model of the MAYBE calculation.

3.4 Simulation Parameters

To simulate probabilistic detection at scales larger than the prototype protocol can create, a larger data set will be extrapolated from the data captured during testing the prototypes. This data set will mimic as closely as possible the results of the protocol if it were to be ran over a longer period of time than is available for this project. The statistical analysis process will also be emulated against the large data set, with all the benefits and limitations of the knowledge as would be experienced in a full implementation with a number of clients.

Firstly, a control group will be created, based on the contents of the MAYBE database during intended operation of the protocol, where an MitM attacker is not present and all parties deposit correct tuples. Analysing the results of the same statistical modeling processes as in the prototype will identify if the patterns and detection heuristics apply at larger scale. This control group will also serve to provide a baseline with which to compare the results of further iterations with alternative datasets.

The data set will then be modified to simulate the results of a pervasive MitM attacker affecting varying percentages of Diffie-Hellman exchanges as seen in the MAYBE database.

As in the control group, the same analysis will be performed on the modified dataset to test the efficacy of detection with different levels of interference from unintended parties. Comparing these results to the results acquired from analysis of the control group, analogous to comparing expected and actual collision value, will illustrate the level of penetration that a MitM attacker can operate at without arousing suspicion through heuristic methods. The hypothesised results are that the greater the penetration of the attacker throughout the data set, the greater the difference between the expected and actual collision value will be.

The control group will then be modified to simulate the case wherein less than 100% of Diffie-Hellman exchanges have both tuples uploaded to the MAYBE database, i.e. a key exchange occurred and only one parties information was uploaded and stored correctly. The hypothesis is that the reduction does not hugely affect the statistical modeling, though it is accepted that some of the "resolution", or the precision of the degree to which the expected and actual collision value differ.

4 Implementation

The prototype MAYBE protocol developed in this project is divided into several interoperating sections, corresponding to different agents in the MAYBE system;

- The MAYBE servers and databases
- Participating servers carrying out Diffie-Hellman exchanges
- Clients in the DH exchange

Each section comprises of various technologies and functional areas, the most important of which are detailed below.

4.1 Deployment Details

4.1.1 MAYBE Server and Database

For both variations of MAYBE, the data was stored and queried in a MySQL database, running on an Apache server using PHP and JavaScript to interface between the client, the web server and the MAYBE web server. This technology stack is a common deployment in the real world, and all the technologies listed are widely used for a variety of applications. PHP and MySQL are known to scale well, allowing the MAYBE database to operate at Internet-scale. The above technologies are also largely platform independent and content delivery over HTTP allows the Diffie-Hellman exchange and MAYBE protocol to operate on a variety of devices and operating systems. Where HTTP, JavaScript and/or PHP are unavailable, a native implementation of MAYBE would be required, though perhaps only in the modules where the provided languages do not function effectively.

Using a combination of HTTP, JavaScript, PHP and SQL, the MAYBE server can accept and process MAYBE tuples from clients and servers, extract and parse the data and add it to the relevant database. The reporting page uses the same technologies to return information extracted from the database in response to queries from the participants.

To simplify installation, the WampServer package [1] was used to create a deployment configuration with close similarity to real-world implementations where WAMP or LAMP (Linux/Unix, Apache, MySQL, PHP) are widely used. Using this kind of package solution is a major factor in convincing end users to participate in the MAYBE protocol, where quick install, minimal configuration and low operation overhead are considered necessary to encourage widespread adoption.

4.1.2 Sample Participant Web Servers

The servers providing the Diffie-Hellman key exchange are running on Apache servers and written in PHP and HTML. Constructing Diffie-Hellman key pairs in PHP was accomplished using the `crypt_DiffieHellman` package from PEAR [5]. This package allowed users to construct entirely new keys and to create key values based on received non-secret data. These calculations are based on integer Diffie-Hellman, though as discussed in section 2.2, the method for creating the secret key is abstract for the purposes of the MAYBE protocol. As long as the key can be treated as an abstract block of data to be calculated in the witness value function, the calculation function, key length, programming language and implementation can be changed as required.

In order to simulate the CallMe version, five web servers were created with separate CallMe identifiers that would accept client requests to perform a Diffie-Hellman key exchange. The client can connect to the server and request the respective server's CallMe and initiate a Diffie-Hellman exchange.

As the server calculates the Diffie-Hellman values necessary to calculate the shared secret and the non-secret values, the server can create the witness value for upload to the database. The non-secret values can then be sent to the client and the Diffie-Hellman exchange continues as normal. The server's side of the MAYBE data can then be uploaded either immediately or stored for later upload as part of a batch operation.

In addition, the MAYBE data is appended to a log file on the web server in a comma-separated value (CSV) format, which can be kept and used to query the MAYBE database through the reporting page.

An alternative way of implementing CallMe requests is to have a well-known uniform resource identifier (URI) as defined in RFC 5785 [20] for web clients to request the CallMe string. This would allow a single URL to be used for clients to request the web server's CallMe string by making an HTTP or HTTPS request for; `https://www.[website-address]/.well-known/get-CallMe`.

4.1.3 Web Clients

To model the heterogeneous nature of devices and communications on the Internet, various clients were used to connect and request DH exchanges under the MAYBE system. These included;

- Laptop running Windows 7 connecting to a locally hosted server
- PC running Windows 7 connecting to a remote server
- Android mobile device connecting to a remote server

The client systems allowed experimentation and testing with a selection of popular web browsers, such as;

- Mozilla Firefox 31.0
- Google Chrome 36.0
- Microsoft Internet Explorer 11.0

Using the above hardware and software configurations, the client connected to a particular web server and requested the web page to begin a key exchange. The server would provide the relevant page containing the information necessary for the client to carry out the Diffie-Hellman calculation and produce the necessary MAYBE information. In both versions of MAYBE, once the key exchange has been completed the client, using embedded JavaScript, uploads their part of the MAYBE tuple pair to the MAYBE database.

The random session MAYBE version has the client decide on a safe prime and a generator and send it to the server over an insecure HTTP POST message. The server then calculates the shared secret based on the received value from the client as well as their non-secret key, which is returned to the client by another insecure HTTP request. The web client can then take the received public key from the server and calculate the shared secret key for the Diffie-Hellman calculation. The client process will the calculate the witness value based on the calculated secret key, choose a value from the random range and upload their tuple to the MAYBE database.

In the CallMe version, the server will provide the CallMe to be used, available on the HTTP page as delivered, and a mechanism for the client to request a Diffie-Hellman key. As in the random session version, the server will calculate and upload their part of the Diffie-Hellman calculation and return the non-secret key to the client over an insecure HTTP form.

The client can then calculate the secret key and continue with the Diffie-Hellman exchange and whichever cryptographic processes it was being used for. The witness value can then be created based on the secret key, appended to the CallMe string and associated meta-data and uploaded to the MAYBE database.

Both versions make extensive use of JavaScript code embedded on the web page, which requires that client's web browser plug-ins and permissions must be set to allow JavaScript to be executed. Knowing the proportion of clients using the web service that allow and disallow scripts becomes very important as this too will affect the ratio of server submitted MAYBE tuples to client submitted MAYBE tuples.

This method of carrying out DH exchanges was chosen to emulate the TLS DH exchange, without having to modify the web browsers, as would be the case if adhering to RFC 5705 [25]. The emulated version has the client choose a prime using JavaScript on the web page and post the DH non-secret data to the server in an HTTP message. In the real world, the browser itself carries out the TLS DH exchange and only a TLS extractor value is used for the MAYBE information.

4.1.4 Changes Required to MAYBE Participants

A key principle guiding the design and development of both versions of the MAYBE protocol was to avoid creating additional requirements on existing processes and services that could participate in exchanges. To achieve this, processes outside of the probabilistic detection were treated as abstract and as far as possible were treated simply as data sources arriving through the MAYBE application programmable interfaces (API).

The MAYBE protocol could be seen from an external developers perspective as a set of APIs or remote procedure calls (RPCs), meaning the developer of an application wishing to take part in the MAYBE service has little input or configuration to do with the MAYBE system itself. As long as the developer can provide a value based on the Diffie-Hellman exchange to use as the seed for the witness value and set random ranges or CallMes as appropriate, the protocol can operate effectively.

In the CallMe version, the web service provider must be able to create or receive the CallMe identifier strings and record how many Diffie-Hellman exchanges take place with a particular CallMe. In addition to this, the server must also keep an approximate record of how many of those clients are likely to have issues uploading their part of the MAYBE tuple, such as those with JavaScript blocking or clients that appear to fail during the exchange process. This record can be useful when comparing the local data with the MAYBE database to identify where suspected client failure caused the tuple pair mismatch, which is separate from a pervasive MitM attack.

4.1.5 Deployment Summary

Figure 8 shows the technical architecture of the MAYBE system. Both the random session and CallMe versions use the same technologies to provide the different functionalities. Many of the technologies outside of the MAYBE server can be replaced without changing the functionality of the system as a whole.

The technologies were chosen to support a level on heterogeneity between instances of the protocol, where different technologies can be used without modification to other major parts of the system. Using HTTP and HTTPS requests for communication between parties allows the scripting languages and database interactions to be changes without modifications to unrelated modules of the MAYBE system.

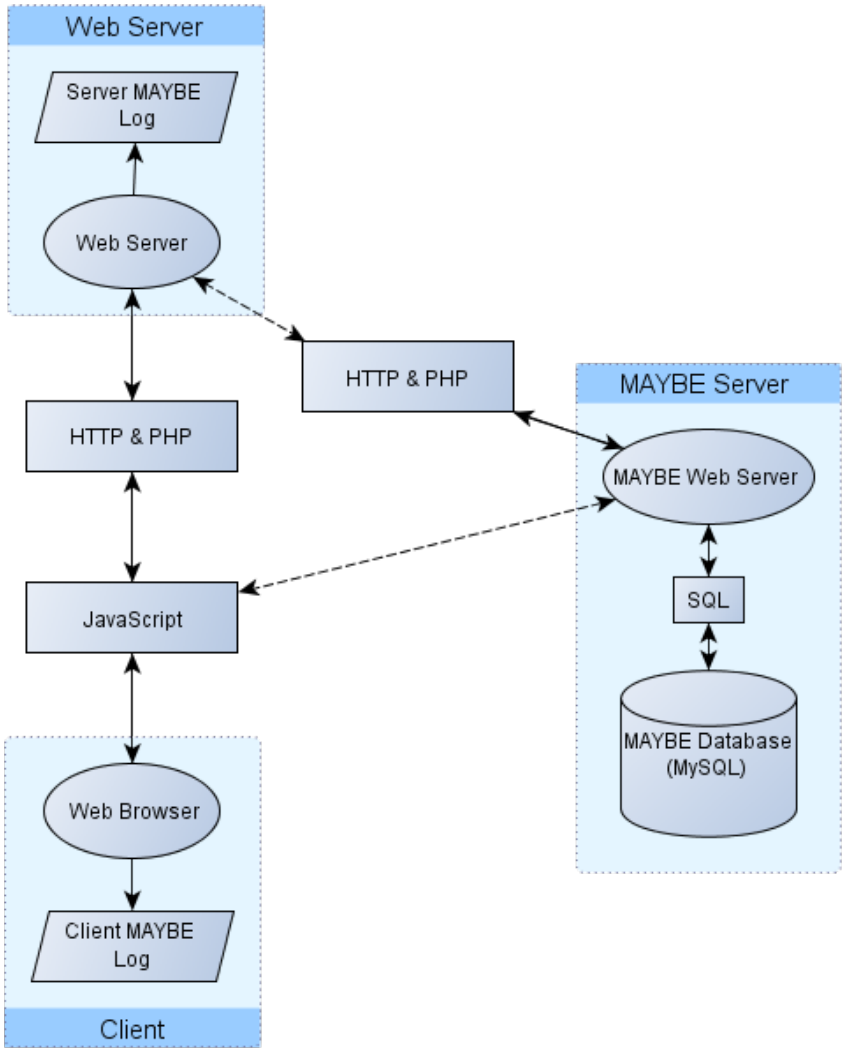

Figure 8: Technical architecture of the CallMe version

For example, the clients and servers in figure 8 could be reprogrammed in Java and communicate using SOAP objects for the Diffie-Hellman exchange, as long as the MAYBE server receives the relevant MAYBE information, the data can be parsed and added to the database.

4.2 Overview of Functional Areas

4.2.1 Database Design

The design of database to store the MAYBE tuples from participants was a key part of the design of MAYBE system as a whole. As sections 3.1.1, 3.1.2 and 3.1.3 show, the database is key to storing and querying the MAYBE data from the participants and for each type of statistical analysis. The database should be designed with the following criteria in mind;

- Independent of data origin
- Able to store large quantities of data
- Efficient to query
- Highly scalable

Using MySQL and Apache servers provides technical solutions for scaling and replicating the MAYBE database, where additional instances of the server and database can be created. Therefore, the database structure must be created in such a way to make updates, querying and management as easy and efficient as possible.

The structure of the database table for the random session version of MAYBE is described as;

```
maybe_list[pk_maybe, session_id, witness_value, upload_timestamp]
```

And the CallMe version;

```
maybe_list[pk_maybe, callme, witness_value, timestamp, from_server]
```

The tables described above are enough for the bare bones operation of each version of MAYBE, but lacks authentication or assigned CallMe functionality. Figures 9 and 10 show the database tables and data types as implemented in the MySQL server.

#	Name	Type	Collation	Attributes	Null
1	**pk_maybe**	int(11)			No
2	**session_id**	varchar(45)	latin1_swedish_ci		No
3	**witness_value**	varchar(45)	latin1_swedish_ci		No
4	**upload_timestamp**	timestamp			No

Figure 9: Database structure for random session version

#	Name	Type	Collation	Attributes	Null
1	**pk_maybe**	int(11)			No
2	**callme**	varchar(256)	latin1_swedish_ci		No
3	**witness_val**	varchar(256)	latin1_swedish_ci		No
4	**timestamp**	timestamp			No
5	**from_server**	tinyint(1)			No

Figure 10: Database structure for CallMe version

Both databases have several fields in common that store data that are necessary for each version of MAYBE;

- `pk_maybe` - The database primary key
- `witness_value` - The witness value as described in section 3.1.6
- `timestamp` - The timestamp associated with the upload of MAYBE data to the database

The purpose of these fields is to increase the complexity of an attacker, not necessarily a man-in-the-middle, to modify or remove tuples from the database without detection. These fields could also be of assistance in detecting if an attacker is interfering with deposits to the MAYBE server before they arrive, as detailed in sections 5.2.1 and 5.2.1.

While the database remains a vulnerable target for any attacker, including MitMs, wishing to cause problems for the operator, the methods of detecting and defending against attacks on the database are better understood and implemented. The MAYBE database operator should investigate any attack on the MAYBE database or its communications, as these will likely be used in tandem with other attacks on the participants.

The fields directly used for the probabilistic detection differ in each version of MAYBE. The `session_ID` in the random session version is used to store the calculated value for each party, while the CallMe version uses `callme`, the received CallMe and `from_server`, a Boolean variable denoting if the tuple was deposited by the server. These values can be queried by the server to detect differences between the results of the expected operation and the operation in the case of a man-in-the-middle.

4.2.2 Reports Created

Depending on the version of MAYBE used and the level of features implemented, different reports will be available form the submitted data. Furthermore, the MAYBE database operator will be responsible for deciding on the accessibility of these reports and the data from which it was produced. Various use cases will demand different handling of access requests; for example a totally public MAYBE database may only allow users to view reports on total data and only query the database for specific witness value and session ID/CallMe pairs.

Once data had been entered in the database, a variety of SQL commands are executed to analyse the patterns in the data and provide a heuristic diagnosis of a pervasive man-in-the-middle. The reports generated contain statistical information such as the total number of records in the database, the number of Diffie-Hellman exchanges seen and the difference between expected and actual hash collision vales. This information can be delivered through an HTML page that can either be posted publicly or held within an organisation running their own MAYBE service.

4.2.3 Comma-Separated Variable Files and Comparison

Each version provides support for participating servers to capture MAYBE tuples as calculated by the server in a log file, stored on their web server for later review and comparison with the data submitted to the central MAYBE database. This allows the server operator to execute a query against the MAYBE database to ensure that the tuples have been correctly deposited and not modified in transit or storage. This is an important part of the detection heuristics as variances in the local log and the database could indicate an attack on the pathway between participating web server and the MAYBE database.

Depending on the storage and deletion parameters of the MAYBE database as defined by the database operator, a scenario could be possible for data to be removed from the database legitimately and then for a query requesting the information to be executed by a third party. This scenario could lead to the incorrect belief that an error is occurring in their upload to the MAYBE database. This occurrence could be ameliorated either through end users of the MAYBE database being aware of the parameters regarding information deletion or through technical means identifying queries for data that has been stored and later expunged from the database.

5 Evaluation

Evaluating the MAYBE system is divided into two sections, covering the different key features of the project.

Firstly the prototype protocol is evaluated, covering the overhead incurred in both communication and storage of operating the protocol and identifying the potential points of failure in the system.

The second part of the evaluation covers the efficacy of the detection metrics based on the available MAYBE data with different data sets, emulating scenarios that occur MAYBE database.

5.1 Evaluation of Prototype Protocol Deployment

5.1.1 Communication Overhead

Random Session

Depending on the configuration of the MAYBE protocol, the tuples will either be uploaded as they are created during the Diffie-Hellman exchange, resulting an additional connection from each participant. Depending on the frequency of Diffie-Hellman exchanges on the clients, this may be negligible, but in cases of high frequency exchanges the usage of secure communication channels may cause unacceptable load.

Using secure transmission methods to upload the data is deemed necessary due to the high importance of ensuring MAYBE tuple information is uploaded without modification. However, the comparison of log data to MAYBE information could be performed over unsecured HTTP in some cases, though HTTPS may be preferred in various use cases.

The MAYBE server will have to be able to accept secure communication channels to receive data from clients, which may require obtaining a certificate allowing them to authenticate themselves to clients. These channels will also require various defensive measures against common attacks such as distributed denial of service and SQL injection attacks.

CallMe

Participants in the Diffie-Hellman exchange will have more choice in when and how to upload tuple information. Participants with a high level of uptime and communication reliability, or clients with a method of storing that data offline until a more opportune moment for uploads can make better use of bulk uploads. Otherwise, single uploads from many users could become computationally intensive, especially at Internet scale, where many connections causing the MAYBE server to hold state, create secure keys and interact with the database result in overload and performance degradation.

5.1.2 Storage Overhead

Random Session

In both instant upload and batch upload, storage is required to record information of the MAYBE exchange for later comparison with database values. This can be relatively small, as data such as the witness value and the session ID can be stored as raw text files, and will be short strings and integers, respectively. Therefore, the size of the files will be highly dependent on the participant's configuration, with more regular uploads reducing the storage size required.

CallMe

Participating servers in this version will need to keep a record of the CallMes they have used and the MAYBE information associated with these CallMes, including the number of exchanges and the witness values associated with each. The server operator should also have an approximation of the percentage of clients that may upload their section of the tuple pairing, based on server-side detection of events that would affect client upload, such as JavaScript errors or terminated connections.

Clients will have a very variable level of storage required, where some may wish to record information of all Diffie-Hellman and MAYBE data exchanges that they have taken part in for future, one-to-one comparison with the online MAYBE database. Others may only store a subset of these to ensure that a sufficient percentage of their tuples are uploaded and stored successfully. Others still may have no interest in later comparing the data and will only hold onto data as long as is required to upload it to the database.

5.1.3 Bottlenecks and Points of Failure

Random Session

The system is highly centralised around the MAYBE web servers and database hosts, where problems with accessibility, response time, security and errors are common. Some of this can be mitigated using distributed system concepts such as replication of the MAYBE database, either in total or in delegation of responsibility of certain subnets. Collaboration or co-operation of MAYBE servers may also improve the resilience and reliability of the protocol, but requires more management and configuration by the system operators.

In certain implementations within organisations where the MAYBE server is kept within the company behind a firewall, the vulnerability is further reduced, though this version of MAYBE is designed to be used in Internet level traffic. The server is still a bottleneck and various methods to protect it still must be taken, though the attack vectors are reduced.

CallMe

While the MAYBE databases are still the bottlenecks in this version of MAYBE, the more distributed approach lowers the demands on each particular server as compared to the random session version.

5.2 Evaluation of Probabilistic Detection Metrics

5.2.1 Control Group

Random Session

In order to simulate a database containing MAYBE tuples from participants, a set of integer values representing the witness values were created, with session IDs selected from the variable random ranges. Once these had been created, the metric as designed were executed against the data with various parameters to identify areas of false positive and negatives.

This data set consisted of 1,100 rows representing 550 Diffie-Hellman exchanges, extrapolated from the data gathered in operating the prototype protocol. While this volume of data is much lower than the expected number of entries in a large-scale MAYBE deployment, this evaluation is focused on the efficacy of the probabilistic detection based on the available MAYBE information.

To create a set of readings to use as a control group, statistical analysis was carried out on a MAYBE data set as would be seen if for each participant in a Diffie-Hellman exchange had contributed a tuple entry, which would correspond to intended operation of the MAYBE protocol. The data set also assumes every party contributing to the database has used the correctly-sized range for the random session value.

Random Range	Number of Tuples	Number of Unique Witnesses	Match Rate (%)	Expected Collision Value	Actual Collision Value	Difference	Avg Diff
3	1100	550	100	183	173	10	
3	1100	550	100	183	185	2	
3	1100	550	100	183	194	11	7.2
3	1100	550	100	183	186	3	
3	1100	550	100	183	173	10	
4	1100	550	100	137	138	1	
4	1100	550	100	137	143	6	
4	1100	550	100	137	136	1	7.8
4	1100	550	100	137	122	15	
4	1100	550	100	137	121	16	
5	1100	550	100	110	124	14	
5	1100	550	100	110	124	14	
5	1100	550	100	110	107	3	10
5	1100	550	100	110	111	1	
5	1100	550	100	110	92	18	
6	1100	550	100	92	100	8	
6	1100	550	100	92	96	4	
6	1100	550	100	92	85	7	6.8
6	1100	550	100	92	86	6	

Figure 11: Subset of the analysis of on the control data set. Each iteration with the same random range has different random session IDs for each witness value, causing varying number of hash collisions

Calculating the standard deviation of the session ID collision rate for different iterations with a particular random range showed useful data regarding cases where outlier collision rates, due to random chance, could be erroneously interpreted as an indicator of an MitM attacker. This sudden raise was often caused by one or more of the iterations having an unusually high or low difference between the expected collision value and the actual collision value, due to the random nature of session ID collision.

CallMe Version

The CallMe version of MAYBE introduces different meta-data and different methods of detecting the presence of a pervasive MitM. The data set used to analyse the detection is similar in many regards to the data set to the random session version, though modified to reflect the different meta-data used.

pk_maybe	callme	witness_value	from_server		Witness_Match?		No. CallMes:	100
551	6999993	1001	0		TRUE		No. Entries	1100
552	444444	1002	0		TRUE		No. DHEs seen	550
553	5555550	1003	0		TRUE			
554	10888878	1004	0		TRUE			
555	8555547	1005	0		TRUE			
556	11111100	1010	0		TRUE			
557	1444443	1031	0		TRUE			
558	9999990	1038	0		TRUE			
559	7555548	1042	0		TRUE			
560	7888881	1053	0		TRUE			
561	10444434	1054	0		TRUE			
562	5777772	1060	0		TRUE			
563	8666658	1079	0		TRUE			
564	10999989	1087	0		TRUE			
565	10999989	1108	0		TRUE			
566	8555547	1115	0		TRUE			
567	6555549	1117	0		TRUE			
568	8555547	1119	0		TRUE			
569	5333328	1127	0		TRUE			
570	6999993	1142	0		TRUE			
571	8777769	1154	0		TRUE			

Figure 12: Subset of the control dataset for CallMe version

This kind of data set could be seen in certain cases where a high rate of tuple matches is expected, especially if very few exchanges take place within time period under investigation. An example of this is communications between MTAs, where transmission links are highly reliable and the servers and services well managed.

5.2.2 Pervasive Man-in-the-Middle - Attempts MAYBE

Random Session

A pervasive MitM that is aware of the existence and operation of the MAYBE protocol may try to upload falsified data to the MAYBE database in order to confuse the heuristic detection algorithms. In order to mask their presence in the a data set, the attacker could upload their version of MAYBE tuples for each party that has been affected. In order to "win", or successfully avoid detection in differences in the expected and actual collision values, the attacker must ensure that the tuples they have uploaded do not cause significant changes in the ratio of expected to actual collision values.

As the attacker is expected to know the random range in use in the particular MAYBE database to which they are attempting to participate with, they can create their own MAYBE data with the witness values from each end point and their chosen random data. However, as the random value is only part of the session ID, the attacker is unable to ensure that they have calculated a "correct" number of guesses.

Detecting this kind of spam attack against the MAYBE database is important, and requires a separate approach from detecting attacks solely on the Diffie-Hellman exchanges. Other heuristics that could be also help to identify this kind of attack include;

- Noticing unusual volumes of traffic from single IP Address ranges
- Unusual patterns of uploads
- Appearing "artificial"; a pattern that is unlikely to appear in operation
- The number of DH exchanges in an interval is much higher than expected

A related issue is where an attacker can block either or both end parties from uploading their respective MAYBE tuples to the correct database. If the attacker is successful in this, they can prevent the affected communications from appearing in the data set and therefore in the heuristic analysis. This is different to an MitM attack, requiring more active attacks on the affected end points, potentially at different times to the original DH exchange. This could be caught through the use of publicly signed certificates on the MAYBE server, where the participating users can verify the identity of the MAYBE server with which they are communicating. Furthermore, a participating client who believes they have uploaded data could query the database for that information and become suspicious if no results are returned.

CallMe Version

For an attacker to avoid detection in the CallMe version, participating in the MAYBE exchange in an attempt to avoid detection is more complex than in the random session version. For an attacker to influence the database in a way to affect the MAYBE comparisons, they have two main options;

- Submit MAYBE tuple pairs with an invented CallMe
- Submitting MAYBE tuples with a stolen CallMe

If the operator knows the CallMes that are being used by legitimate users of the MAYBE system, they can quickly discover any tuples that do not match the CallMes from these users. Unknown CallMes could identify issues such as unregistered servers submitting to the database, incorrectly configured participants or an attacker depositing fraudulent information to affect the detection metrics. This method is most effective in implementations where the participants have a high likelihood of depositing each tuple and relatively few CallMes are used.

Finding these tuples could indicate any of;

- Supernumerary parties in an Diffie-Hellman exchange
- An attacker impersonating a server and submitting data to the database
- Fault in configuration or auditing logs

Deploying a MAYBE implementation with knowledge of all used CallMes is difficult at Internet scale. Complex issues that must be addressed include the balance of privacy to authentication, the distribution of the MAYBE servers and methods of creating and issuing secure CallMes.

Despite issues of deployment at Internet-scale, independent organisations may be able to manage this kind of infrastructure, allowing them to manage the MAYBE configuration for their own networks.

5.2.3 Pervasive Man-in-the-Middle - Avoids MAYBE

Random Session

In contrast to where an attacker attempts to directly affect uploads to the MAYBE database, an attacker could avoid the MAYBE server and decline to upload tuple data for the exchanges they have affected. While an intelligent adversary is more likely to at least attempt to modify their signature on the data set, legacy implementations of MitM attacks could be caught, forcing agencies to withdraw and redesign existing attacks.

To investigate the efficacy and resilience of MAYBE to MitMs avoiding participation, analysis was then carried out on the efficacy and rate of correct detection of a pervasive Man-in-the-Middle.

The control data set was modified to represent the outcome if an MitM attacker had been present during the Diffie-Hellman exchanges and had not submitted data to the MAYBE database.

To evaluate the efficacy of detecting interference from a pervasive MitM by calculating the the actual and expected collision values, the MAYBE heuristics were performed on data sets with varying levels of interference from an MitM. Each level of MitM interference was calculated with random ranges of 5, 10 and 15, with five iterations of each, with differing session IDs for each witness value.

The difference between expected and actual collision values becomes more pronounced as the ratio of attacked transmissions increases against the number of unaffected key exchanges. After calculating the absolute difference between the expected and actual collision value, this figure can be divided by the expected collision value to identify the degree to which they differ. A sample of the results can be seen in table 4.

% MitM-ed	Expected CV	Actual CV	Difference
0%	37	35	4.3%
20%	37	24	34.6%
30%	37	19	49.2%
50%	37	14	62.7%
80%	37	6	82.7%

Table 4: The results of the MAYBE expected and and actual collision value calculations based on the control data set as modified with varying levels of interference from an MitM

CallMe Version

Detecting an MitM attack from records in the database for the CallMe version is slightly different to the random session version, as it benefits from additional data to make better decisions. A pervasive man-in-the-middle attacker can have the following signatures as seen from database queries;

- Removed tuples - If a participant compares the tuples in the database to the Diffie-Hellman exchanges it participated in and uploaded to the MAYBE server, differences could indicate an MitM attacker preventing or modifying tuple uploads.
- Entry from client only - An entry from a client that does not have a matching deposit from a server could indicate an MitM affecting the shared secret and therefore the witness value, issues with the server uploading to the MAYBE database or a spam attempt on the database.
- Unrecognised CallMe string - While this requires knowledge of the expected CallMe strings in the data set, any string that does not match expected values could indicate either an error in the configuration of a MAYBE participant, or an attacker forging CallMe strings.
- More tuples than can be accounted for - An attacker that has attempted to upload their MAYBE tuple data as well as the client, results in more tuples than would result from normal interaction. This could be at micro scale, where more tuples exist than should exist for a particular CallMe, or at a macro scale, where there are more tuples in a data bset than can be accounted for.

Depending on the meta-data available in the instance of the MAYBE installation, some of the listed detection methods may or may not be available.

Comparing local data with the MAYBE database

The expected use case of the CallMe version is for servers or clients providing data to the database to be able to search for the data they have uploaded to verify that both a matching tuple from the intended other party has been uploaded and their tuple has been stored without modification. Participants can query the database for entries matching the CallMes they used with and receive the tuples matching that value. Comparing these to the Diffie-Hellman exchanges that they have records of actively participating in can identify any instances where either an attacker has been imitating the server's CallMe or there are an unusual amount of client responses.

To simulate this, a participant's MAYBE data for a series of Diffie-Hellman exchanges is created and compared with the data in the database for any inconsistencies between the two. Modifying a tuple in the database to cause a mismatch in the witness value/CallMe pairing will imitate an exchange that had interference from an MitM. The client's comma-separated variable file represents four Diffie-Hellman exchanges with different CallMe strings for each. The client can search for the matching tuples from the server, based on either or both of the values stored. Searching by the CallMes only returns all exchanges carried out for each CallMe value, which returns data about all users of the system, which is not desirable in all cases. Searching by both values only returns any tuples that match both and therefore only the tuples that that user has participated in, assuming they have been uploaded by the server without interference of an MitM.

Depending on the configuration of the database, the participant may be forced to upload both the CallMe and witness value in a query and only receive results that match both values, so as to preserve privacy of other users in the database. This search could be done with a lower tolerance value as the client will likely know if they have submitted their tuples, so any mismatch could be suspicious.

callme_hash	witness_value
7999992	1117
2888886	1219
8555547	1339
4555551	1507

Figure 13: Search results for a witness value/CallMe pair, as would be seen by a client searching for their own data

Figure 13 shows the results of searching for the witness value and CallMe pairings from one client against the databases where the protocol worked as intended with no MitM affecting the exchange at either end in any transaction. Any modification to either the CallMe or witness value causes a mismatch, and the affected results to not be returned. In this case, further heuristic analysis could be carried out, for example checking that the requested information has not been removed from the database due to expiry or the server has simply not uploaded it yet. Alerts could also then be raised to the MAYBE database operator and the server administrator for any affected tuples, informing them of the error, along with any pertinent information.

Fewer Tuple Pairings

Depending on the underlying protocol, there will be variations in the number of tuples that have a corresponding entry representing the other side of the Diffie-Hellman exchange.

In order to simulate fewer tuple pairings than in the control dataset, a subset of the data is deleted, along with related meta-data such as the primary key. This modified data set is then analysed to identify the hash collision rate, which will be much lower, and how the probabilistic detection functions with fewer legitimate hash collisions. In the MAYBE database, this will be noted as there being fewer witness value pairings than would otherwise be expected, resulting in a lower total amount of tuples in the database as well as a lower results from an SQL query to count the witness value matches.

As the expected collision value is based on the number of complete tuple pairings, the lack of a certain number of matches does not affect the detection significantly. However, the reduced number of tuple matches does reduce the fidelity of the probabilistic detection by lowering the number of tuples that can be considered for the expected collision value. Depending on the underlying protocol, the amount of incomplete tuple pairings may be cause for concern, though not a direct indicator of an MitM over the data set.

5.3 Evaluation Summary

The evaluation shows that while both versions of MAYBE are effective, the CallMe version can be considered better suited to the requirements of Internet-scale deployment, due to its more hierarchical and distributed nature. Ultimately, both versions serve to provide probabilistic detection of MitM attacks and the version used will depend on the operator's choice based on the underlying protocol and the nature of the participating clients.

An operator wishing to run a MAYBE service must choose the version and the parameters that best meets their requirements in regard to storage space, management overhead and participant willingness to provide information.In basic terms, the random session version requires less management overhead on operators and participants, while the CallMe string version requires more meta-data collection and modification to participants.

6 Conclusion

The need for techniques and technologies to detect a Man-in-the-Middle attacker across large volumes of data has been illustrated throughout this project. Each version of MAYBE described, designed and evaluated show potential for post-facto detection of a pervasive MitM attack in large volumes of data. While the prototype MAYBE implementation is limited in regards to the security of uploads and authentication, the preliminary results indicate that the MAYBE statistical modelling can expose the existence of an MitM. A deployment based on the prototype protocol could be used in smaller implementations in corporate or organisation-size networks, though unable to scale to Internet levels of traffic.

As an agency engaged in pervasive monitoring will likely try and avoid being detected by any methods, so the counter-measures they might use and the signatures of which must be understood. In many security spheres, an "arms-race" scenario is encountered, where each development in security measures is matched by attacks and vulnerabilities. This kind of scenario is highly likely in the MAYBE deployment, where the adversary is intelligent, well-funded and wishing to avoid detection. Therefore, understanding the possible methods by which an attacker could attempt to interfere with the detection metrics over time will be important during the deployment of MAYBE.

MAYBE could also be useful in acting as a "canary", where even if the data cannot lead to a conclusive finding about the existence of an MitM, the fact that a "doubt" exists could alert the operator to take further action.

Limitations exist in the deployment of each version of MAYBE. A level of standardisation would be needed, with a level of management overhead in delegation and management. In both versions, having a concrete well-known URI for various parts of the intercommunication in the system could allow for greater deployment, due to the decoupling of some of the message passing from the calculations. Further integration with the web browsers is needed, and the deeper integration with other technologies that could take place in the MAYBE system, in order to reduce the barriers to entry for participants. An example of this kind of integration could be creating an extension for open-source browsers like Mozilla Firefox or Google Chrome, taking TLS extractor values from the browser's DH exchange and using that for MAYBE information.

Further research and development in this area would be beneficial before a major deployment. Topics of research could consider topics such as;

- Heterogeneous implementations of MAYBE participants contributing to a single MAYBE server
- The use of public key cryptography to create CallMes
- Different topologies for the distribution of hardware and software resources

Particularly in the CallMe version, understanding the benefits and characteristics of different technical and functional infrastructure deployments as they relate to hierarchical structures of CallMe and heuristic analysis would be beneficial. Deploying CallMes, and the analysis thereof, in a hierarchical manner could reduce the computational and storage overhead on individual servers.

Overall, the MAYBE system has potential in detecting a pervasive man-in-the-middle attack against a number of Diffie-Hellman exchanges, though further work is needed before mass deployment.

References

[1] WampServer Official Website. http://www.wampserver.com/en/. [Online; accessed 24-June-2014].

[2] AT&T's role in dragnet surveillance of millions of its customers. https://www.eff.org/files/filenode/att/presskit/ATT_onepager.pdf, 2013. [Online; accessed 17-May-2014].

[3] Jewel v. national security agency, 2013. Case No. 3: 08-cv-04373-JSW.

[4] Shahram Bakhtiari, Reihaneh Safavi-Naini, Josef Pieprzyk, et al. Cryptographic hash functions: A survey. *Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Australia*, 1995.

[5] Pádraic Brady and Bill Shupp. Implementation of Diffie-Hellman Key Exchange cryptographic protocol for PHP5l. http://pear.php.net/package/Crypt_DiffieHellman/, 2012. [Online; accessed 08-June-2014].

[6] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security and Privacy*, 7(1):78–81, 2009.

[7] Bailey W Diffie, Martin E Hellman, and Ralph C Merkle. Cryptographic apparatus and method, 1980. US Patent 4,200,770.

[8] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[9] Dave Emery, Steve Bellovin, and Neil King. NSA Fibretap. http://cryptome.org/nsa-fibertap.htm, 2001. [Cryptome; Online; accessed 13-May-2014].

[10] Stephen Farrell. A Man-in-the-Middle Detection Protocol (MAYBE?). 2014.

[11] Ryan Gallagher. New Snowden documents show NSA deemed Google networks a "Target". http://www.slate.com/blogs/future_tense/2013/09/09/shifting_shadow_stormbrew_flying_pig_new_snowden_documents_show_nsa_deemed.html, 2013. [Online; accessed 12-May-2014].

[12] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[13] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[14] IS ISO. 7498. *Reference Model for OSI*, 1982.

[15] Aleksandar Jurišic and A Menezes. Elliptic curves and cryptography. *Dr. Dobb's Journal*, pages 26–36, 1997.

[16] Vivek Kapoor, Vivek Sonny Abraham, and Ramesh Singh. Elliptic curve cryptography. *ACM Ubiquity*, 9(20):20–26, 2008.

[17] Kristin Lauter. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications*, page 63, 2004.

[18] Ewen MacAskill, Julian Borger, et al. GCHQ taps fibre-optic cables for secret access to world's communications. http://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa, 2013. [The Guardian; Online; accessed 13-May-2014].

[19] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology—CRYPTO'85 Proceedings*, pages 417–426. Springer, 1986.

[20] Mark Nottingham and Eran Hammer-Lahav. Defining well-known uniform resource identifiers (URIs). 2010.

[21] ASK Pathan, Hyung-Woo Lee, and Choong Seon Hong. Security in wireless sensor networks: issues and challenges. In *Advanced Communication Technology, 2006. The 8th International Conference*. IEEE, 2006.

[22] Martin Petraschek, Thomas Hoeher, Oliver Jung, Helmut Hlavacs, and Wilfried N Gansterer. Security and usability aspects of man-in-the-middle attacks on ZRTP. *J. UCS*, 14(5):673–692, 2008.

[23] Bart Preneel. Cryptographic hash functions. *European Transactions on Telecommunications*, 5(4):431–448, 1994.

[24] NIST FIPS PUB. 180-4 secure hash standard, march 2012.

[25] Eric Rescorla. Keying material exporters for transport layer security (TLS). 2010.

[26] Ronald L Rivest and Adi Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–394, 1984.

[27] Katitza Rodriguez. Looking back one year after the Edward Snowden disclosures - an international perspective. `https://www.eff.org/deeplinks/2014/05/looking-back-one-year-after-edward-snowden-disclosures-international-perspective`, 2014. [Online; accessed 17-May-2014].

[28] Koen Rouwhorst. FLYING PIG: GCHQ's TLS/SSL knowledge base. `http://koen.io/2013/12/flying-pig-gchq-tls-ssl-knowledge-base/`, 2013. [Online; accessed 14-May-2014].

[29] Bruce Schneier, Brian Carpenter, and Stephen Farrell. IETF 88 Technical Plenary: Hardening The Internet. `https://www.youtube.com/watch?v=oV71hhEpQ20`, 2013. [Online; accessed 13-May-2014].

[30] David Terr. Diffie-Hellman Protocol. `http://mathworld.wolfram.com/Diffie-HellmanProtocol.html`, 2012. [Online; accessed 19-May-2014].

[31] S Turner and L Chen. RFC 6151-Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms, 2011.

[32] Yang Zhang, Nirvana Meratnia, and Paul Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE*, 12(2):159–170, 2010.

[33] Philip Zimmermann, Alan Johnston, and Jon Callas. ZRTP: Media path key agreement for secure RTP. *draft-zimmermann-avt-zrtp-04 (work in progress)*, 2007.