What vulnerabilities and threats exists for public-key cryptography?

Tomas Melin
Linköping University
Linköping, Sweden
tomme578@student.liu.se

Tomas Vidhall
Linköping University
Linköping, Sweden
tomvi780@student.liu.se

ABSTRACT
[Awesome Text]

Keywords
public key cryptography, Diffie-Hellman key exchange, asymmetric cryptography

1. INTRODUCTION
The public-key cryptography, or asymmetric encryption as it is also referred to, is a procedure with the main objective to ensure privacy or confidentiality. The procedure uses two keys, public and private key. Each person has a public key that is used to encrypt messages and can be shared in any possible way. Each user also has a private key which is only known to that specific user. If a user recieves a message that is encrypted with his public key he can decrypt it with his private key and no one else can decrypt it.
This can be explained further using mathematical notations. An entity has a public key e and a corresponding private key d. For a system to be secure, it has to be fairly impossible to compute d, given e. The public key defines an encryptation transformation E_e and the private key defines the associated decryptation transformation D_d. This means that if an entity B wants to send a message to A, it has to retreive a copy of A's public key e and use E_e to obtain the ciphertext $c = E_e(m)$, where m is the message in plaintext, and send c to A. Once A has received the ciphertext c, A uses D_d with the private key d to decrypt the cipher text into the original message in plain text $m = D_d(c)$. [1]
By the above description it is clear that using public-key encryption on its own, does not provide data origin authentication, due to the fact that A's encryption transformation is public information. Data origin authenticication can be described as a mechanism where the data is authenticated. [1]

1.1 Motivation
The concept with private and public keys is very fascinating to us and we also know that it is impossible to read messages which have been encrypted with a foreign public key. This makes us motivated to study the public-key exchange more carefully and investigate the subject thoroughly. We plan to find the advantages and the disadvantages of using the public-key exchange and how it is possible to take advantage of those, in order to 'crack' the implementation.

1.2 Purpose
We want to extend our knowledge in the subject of communicating over a unsafe network using the public-key exchange. We also would like to investigate how safe the public-key exchange is and what threats that has been a vulnerability, and what threats that still exists.

1.3 Questions of Interest
- How does public-key cryptography work?
- How has public-key cryptography implementations changed through history?
  - Has it been affected by cracked implementations?
- What attacks are public-key cryptographys vulnerable to?
  - What weaknesses can a Man in the Middle-attack take advantage of?
  - Is it possible to prevent these attacks through authentication?
- How does the Diffie-Hellman key exchange work?
  - What attacks is the Diffie-Hellman public-key exchange vulnerable to?
  - What is the benefit of using symmetric cryptography compared to public-key cryptography?

1.4 Limitations
When first choosing this subject we thought that we would be able to simulate and test the public-key cryptography, however once we started to investigate what had been done in other research papers we decided that we would only do a theory based investigation and at most a small experiment to simulate and its properties.

2. THEORY

There have been several public-key encryption schemes throughout the history but the most significant ones are RSA, Rabin, ElGamal, McEliece, Merkle-Hellman knapsack, Probabilistic public-key encryption. The procedure which these encryption schemes use to generate their key is different from scheme to scheme but we will show how a number of those generate their key.

2.1 Public-key encryption schemes

2.1.1 RSA public-key encryption

which is short for R. Rivest, A. Shamir, and L. Adleman and is the most widely used public-key cryptosystem. [1] RSA's security is based on the intractability of the integer factorization problem, that is, given a positive integer n, your task is to find its prime factorization.

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

Where p_i are distinct primes and each $e_i \leq 1$. [1] The key generation for RSA public-key encryption procedure containing five steps. Each entity is required to have RSA public key and a corresponding private key.

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$
3. Select a random integer e, $1 < e < \phi$, such that $gcd(e, \phi) = 1$, where $gcd(x, y)$ is the greatest common divisor for x and y.
4. Use the extended Euclidean algorithm to compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 (\mathrm{mod} \phi)$
5. The entity's public key is (n, e) and the private key is d [1]

The algorithm for RSA public-key encryption has two phases, encryption and decryption, as described in [1] on p.286.

Encryption In this case, B encrypts a message m for A, which A decrypts.

1. Obtain A's authentic public key (n, e).
2. Represent the message as an integer m in the interval $[0, n-1]$.
3. Compute $c = m^e \bmod n$, using the extended Euclidean algorithm.
4. Send the cipher text c to A.

Decryption To recover plain text m from c, A should do the following:

1. Use the private key d to recover $m = c^d \bmod n$.

This version of RSA is what is called the textbook version of RSA, i.e. the version that is taught but is not actually used. The difference between this and the version of RSA that is being used in the real world is that the non-textbook version uses what is called *padding*. Padding can be explained as to insert a structured randomized sum of characters into the message m, which is to be encrypted. This is done in order to separate e.g. two equal messages (m_1 and m_2) and their corresponding cipher texts (c_1 and c_2) from having similar cipher texts, thus avoiding that a potential attacker can distinguish which cipher text represents which plain text message.

2.1.2 Rabin public-key encryption

is the first example of a provably secure public-key encryption scheme. Many cryptosystems, for example RSA, are believed to be as hard to break as the mathematical problems they build on, but few can really prove it. Rabin is one of those where it is proven to be as secure as the related mathematical problem, the integer factorization problem. Rabin works just as RSA except for differences in key-generation. Each entity A is required to have both a public and a private key and they are generated according to the following steps.

1. Generate two large, random, different prime numbers p and q, each roughly the same size.
2. Compute $n = pq$.
3. A's public key is n and A's private key is (p,q).

Case B encrypts a message m for A which A then decrypts.

Encryption What B should do.

1. Obtain A's public key n.
2. Represent the message as an integer m in the range {0,1,...,n-1}.
3. Compute $c = m^2 \bmod n$.
4. Send c to A.

Decryption What A should do.

1. Use the algorithm to find square roots modulo n given its prime factors p and q. This gives you 4 messages m_1, m_2, m_3 and m_4.
2. A somehow decides which of these possible messages is m.

2.1.3 Merkle-Hellman knapsack encryption

is the first concrete realization of a public-key cryptosystem. It is based on the mathematical problem called the *super-increasing subset sum problem*. It differs from RSA through the cryptation being only one-way. The public key is only used for encryption and the private key is only used for decryption. You can not get a message encrypted with the private key and decrypt it using the public key. Therefore it can't be used for cryptographic signing. Key generation uses the following steps.

1. An integer n is fixed as the length of the public key.
2. Choose a superincreasing sequence $(b_1,b_2,...,b_n)$.
3. Choose a integer M such that $M > b_1+b_2+...+b_n$.
4. Choose another integer W such that $1 \leq W \leq M-1$ and $\gcd(W,M) = 1$.

5. Select a random permutation π of the integers {1,2,...,n}.
6. Compute $a_i = Wb_{\pi(i)} \bmod M$ for i = 1,2,...,n.
7. The public key is (a_1,a_2,...,a_n) and the private key is (π,M,W,(b_1,b_2,...,b_n)).

Case B encrypts a message m for A which A then decrypts.

Encryption What B should do.

1. Obtain A's public key (a_1,a_2,...,a_n).
2. Represent the message as binary string of length n, $m = m_1m_2...m_n$.
3. Compute the integer $c = m_1a_1 + m_2a_2 + ... + m_na_n$.
4. Send c to A.

Decryption What A should do.

1. Compute $d = W^{-1}c \bmod M$.
2. By solving a superincreasing subset sum problem using the algorithm, find integer r_1,r_2,...,r_n. $r_i \in \{0, 1\}$, such that $d = r_1b_1 + r_2b_2 + ... + r_nb_n$.
3. The message bits are $m_i = r_{\pi(i)}, i = 1, 2, ..., n$.
4. $m = m_1m_2...m_n$

2.1.4 ElGamal public-key encryption

also known as the Diffie-Hellman key exchange. The security of ElGamal is based on the intractability of the discrete logarithm problem and the Diffie-Hellman problem, as [1] wrote on p.294. Two versions of the ElGamal public-key encryption will be described, the basic and the generalized version. The key generation for the basic ElGamal is executed in the following steps:

1. Generate a large random prime p and a generator α of the multiplicative group $\mathbb{Z}_p^*$ of the integers modulo p using algorithm 4.84.
2. Select a random integer a, $1 \leq a \leq p-2$, and compute $\alpha^\alpha \bmod p$ using algorithm 2.143.
3. Which gives the following: A's public key is $(p, \alpha, \alpha^\alpha)$ and A's private key is α.

Basic ElGamal public-key encryption The next part describes how the basic ElGamal encryption and decryption functions, in the case where B is to encrypt a message m for A, which is to be decrypted by A, according to p.295 in [1].

Encryption In order to encrypt the message m, B has to complete the following steps:

1. Obtain A's authentic public key $(p, \alpha, \alpha^\alpha)$
2. Represent the message as an integer m in the range $\{0, 1, ..., p-1\}$
3. Select a random integer k, $1 \leq k \leq p-2$.
4. Compute $\gamma = \alpha^k \bmod p$ and $\delta = m\dot{(}\alpha^\alpha)^k \bmod p$
5. Send the cipher text $c = (\gamma, \delta)$ to A.

Decryption To recover the plain text m from the cipher text c, A should proceed with the following steps:

1. Use the private key a to compute $\gamma^{p-1-a} \bmod p$
2. Retrieve m by computing $(\gamma^{-a}) \cdot \delta \bmod p$.

The generalized version of ElGamal encryption is based on the generalization of the basic version, i.e. the multiplicative group $\mathbb{Z}_p^*$ can be generalized to work in any finite cyclic group G, according to [1] on p.297. The group G should be carefully chosen to fulfill the following conditions:

1. for efficiency, the group operation in G should be relatively easy to apply, and
2. for security, the discrete logarithm problem in G should be computationally infeasible.

The key generation for the generalized version of ElGamal can be described in three steps, where each entity A should perform each step, after obtaining a public key and a corresponding private key:

1. Select an appropriate cyclic group G of order n, with generator α, assuming that G is written multiplicative.
2. Select a random integer a, where $1 \leq a \leq n-1$, and compute the group element α^α.
3. A's public key is (α, α^α), together with a description how to multiply the elements in G and A's private key is a.

Generalized ElGamal public-key encryption B encrypts a message m for A, which A decrypts.

Encryption B should do the following:

1. Obtain A's authentic public key (α, α^α).
2. Represent the message as an element m of the group G.
3. Select a random integer k, $1 \leq k \leq n-1$.
4. Compute $\gamma = \alpha^k$ and $\delta = m \cdot (\alpha^\alpha)^k$.
5. Send the cipher text $c = (\gamma, \delta)$ to A.

Decryption To recover the plain text m from c, A should do the following:

1. Use the private key a to compute γ^a and then compute γ^{-a}.
2. Recover m by computing $(\gamma^{-a}) \cdot \delta$.

2.1.5 McEliece public-key encryption

The McEliece public-key encryption scheme is based on the difficulty of decoding a general linear code, i.e. an error correcting code where any linear combination of code words, is also a codeword. [**?**] The idea behind this scheme is to first select a particular code for which an efficient decoding algorithm is known, and then to disguise the code as a general linear code. Due to the fact that the procedure of decoding an arbitrary linear code is NP-hard, a description of the original code can serve as the private key and a description of the transformed code serves as the public key, as written on p.298 [1].

2.1.6 *Probabilistic public-key encryption*

Probabilistic encryption can be described as using randomness to achieve a provable and very strong level of security, according to p.306 [1]. This is used to prevent the occurrence of equal cipher texts when encrypting the same plain text message several times.

2.2 Authentication

Public-key encryption sounds really great in theory and the math checks out. However there are of course issues with it. The biggest issue is authentication. If someone sends you their public key, how do you know that they are who they say they are? A man in the middle attack is an example of this problem.

2.2.1 *Man in the middle attack*

A man in the middle attack is an attack which lives on the problem of authentication. An example would be that you are requesting the public key of Google with a message to their address. The attacker catches the message and sends back his public key instead of Google's. Now you will of course assume that you have Google's public key and can send encrypted messages to them. What can happen is that the attacker will just take your packets, read them, send them to Google, read their response and send them back to you. He can do all of this without leaving any traces and neither you nor Google will know that he could read your "encrypted" information. He can also pretend to be Google and read your messages and send responses in their name. To send secure encrypted information we need some sort of authentication to prove you are who you say you are.

3. METHOD

Initially we performed a literature study in order to find sources of relevant information for our report. We did this by searching online and in the recommended literature on the course homepage. Once we found literature which appealed to us, we analyzed the material to find out if it was legitimate to use in our report. In the following weeks we proceeded by structuring our report in sections which were recommended on the course web page.

4. RESULTS

Due to the fact that encryption transformation is public information, leads to the lack of data origin authentication in public-key encryption, where data origin authentication can be described as a mechanism where the data is authenticated and proven to legitimate. The next sections describe what available solutions exists.

4.1 Certificate authorities

In order to authenticate that a website is who it say it is, the currently used solution is to use certificates. A certificate normally contains a number of fields, such as:

- id,
- subject,
- what signature algorithm has been used to create
- the signature,
- signature,
- issuer,
- valid-from,
- valid-to,
- what purpose the key was intended for,
- public key, fingerprint algorithm,
- fingerprint.

4.2 Bitcoin

About bitcoin...

4.3 Web of Trust

5. CONCLUSIONS

[Awesome Text]

6. DISCUSSION

Idea: By the above description it is clear that using public-key encryption on its own, does not provide data origin authentication, due to the fact that A's encryption transformation is public information. Data origin authentication can be described as a mechanism where the data is authenticated.

7. REFERENCES

[1] v. O. S. V. A. Menezes, P. Handbook of applied cryptography. 1996.