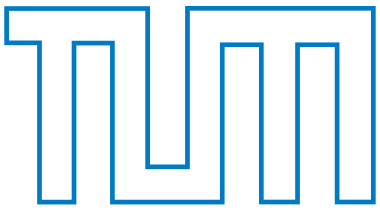


Network Security

Summary

Author: Thomas Pettinger

2017–03–02

Network Security
Technische Universität München

Table of Contents

1 Introduction

1.1 Attacks and Attack Detection

Attacks can have different impacts on the target. Disruptive attacks try to fully deny the service (DoS) of the victim whereas degrading ones only occupy parts of the resources. A DoS attack can also executed distributed, a so called DDoS Attackers might also try to gain confidential data or control the target system. Port scans can be used to gain information about the network topology, operating systems and applications or application versions.

To be able to tell if a system is under attack, different measures can be taken at different points in the system.
Host intrusion detection systems (HIDS) are located on the host system. This enables easy detection using information available on the potential victim system but it has to be present on every system (expensive deployment) and the attack actually reaches the victim and is not detected in advance. Network intrusion detection systems (NIDS) lay on the network layer which enables the detection of attacks before they reach the host.
One of the detection methods available is knowledge-based detection. Known signatures of attacks are compared to the actual traffic and if the patterns match an alarm is raised. This only detects known attacks though. To improve this shortcoming, anomaly detection in traffic, protocol or applciaiton behavior can be used. Anomalies can be detected with different metrics in mind. A very simple one might be the number of request, but this does not take legitimate change in traffic into account. A better approach is using cumulative sums which are low if the average is small or whenever only small amounts of values are large but grows if the amount of large values grows in a certain point in time. The disadvantage of anomaly detection though is that oftentimes the rate of false-positives is high.
Detecting attacks is not easy and network monitory often comes at a cost. It is important though especially in large systems when the attack surface grows. The challenge is to find a good compromise between security and performance.

1.2 Attacker Model and Locations

We generally assume the attacker to be (in) the network. They can perform any active or passive attack but cannot break cryptographic primitives. Active attacks are attacks where some influence is measurable e.g. a delay, modifications or replays whereas a passive attack simply stands for eavesdropping messages. This model is called the Dolev-Yao attacker model.

Attackers can be located on different parts of the network and depending on this location different attacks are possible. If the attacker is close to you they are able to perform active attacks like message modifications on you. This can be circumvented by communicating over a secure tunnel though. If the attacker is close to your servers, timing attacks are possible where attackers can measure how long certain operations take to break into the system. The last possibility is that the attacker is somewhere in the Internet. Since the end user has no control over how packets are routed, attackers can modify the path they take for example.

1.3 Security Goals

Data Integrity No improper or unauthorized change of data

Confidentiality Concealment of information

Availability Services should be available and function correctly

Authenticity Entity is who she claims to be

Accountability Identify the entity responsible for any communication event

Controlled Access Only authorized entities can access certain services or information

1.4 Threads

We define a thread in a communication as any possible event or sequence of actions that might lead to a violation of one or more security goals. The actual realization of a thread is then called attack.

Different threads are possible:

- **Masquerade**: An entity claims to be another entity (also called "impersonation")
- **Eavesdropping**: An entity reads information it is not intended to read
- **Loss or Modification of (transmitted) Information**: Data is being altered or destroyed
- **Denial of Communication Acts (Repudiation)**: An entity falsely denies its participation in a communication act
- **Forgery of Information**: An entity creates new information in the name of another entity
- **Sabotage/Denial of Service**: Any action that aims to reduce the availability and / or correct functioning of services or systems
- **Authorization Violation:**: An entity uses a service or resources it is not intended to use

2 Language-theoretic Security

Communication protocols define the procedure and the format of exchanged messages. If two communication partners speak the same protocol it is not necessarily given that they have the same understanding though. Different problems may arise which result in some guidelines to avoid them:

Full input recognition Programmers usually assume well formed input where in reality the input is controlled by the attacker. For this reason every input should be checked if it fulfills the expectations of valid inputs.

Only Type 2 or 3 of Chomsky Hierarchy grammars for inputs Do not define turing-complete protocols because recognition of the input and testing the equivalence of implementations is undecidable.

Grammar	Language	Recognized by
Type 3	Regular	Finite state automaton
Type 2	Context-free	Pushdown automaton
Type 1	Context-sensitive	Some weird stuff
Type 0	recursively enumerable	Turing machine

Figure 1: Chomsky Hierarchy of Languages

Reduce computing power Reduce the computing power exposed to the outside. Computing power that is not there can not be exploited. This also includes defining protocols only in type 2 or 3 languages since parsing types 0 and 1 requires large or potentially unlimited amounts of computing.

Same interpretation of messages All participants of a communication should interpret messages the same way. For this parsers have to be equivalent which is only decidable for type 2 and 3.

3 Firewalls and Security Policies

We define a system as secure if, started in an allowed state, always stays in states that are allowed.

3.1 The three Security Components

- **Requirements** define security goals
- **Policy** are rules to implement the requirements
- **Mechanisms** enforce the policy

3.2 Network Firewalls

Firewalls provide **controlled access** at the network level to resources. They are usually deployed where a protected subnet is connected to a less trusted network like the internet (c.f. Figure 2). From the firewall's point of view, there are incoming and outgoing packages on all interfaces. These can be filtered using two different strategies.
Whitelisting, which is considered as the best practice, has the default behavior of denying all packets and allowing only explicitly permitted ones whereas **blacklisting** allows everything except blocked packets.

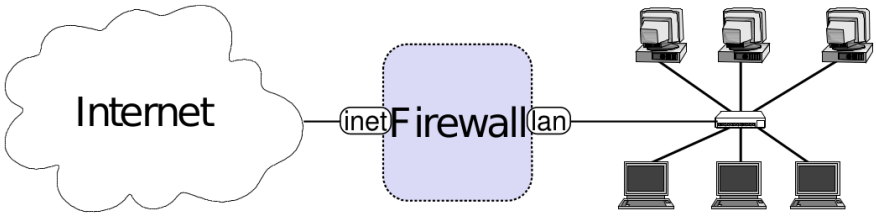


Figure 2: Firewall Placement

Configuring Firewalls

Firewalls are configured with rulesets which are traversed sequentially until a matching one is found. Those rules are composed of a matching condition and an action. Possible actions are for example accept, drop, reject or log. The matching conditions are a bit more complex. They include the incoming interface, several layer 2 to 4 packet fields (srcMAC, dstMAC, srcIP, dstIP, prot, srcPort, dstPort, flags,...), states (in case of stateful matching) and other relevant conditions.

Stateful Firewalls

Stateful Firewalls keep states of connections with the help of IP-5-Tuples (srcIP, dstIP, protocol, srcPort, dstPort). A new state is generated when a packet with a new tuple arrives which transitions from NEW to ESTABLISHED state. It is usually advisable to put more frequently used rules to the top of the firewall, i.e. matching established connections (new connections are rarer) and to check that ports are above the well defined port range (≥ 1024). Figure 3 shows an example of a stateful firewall configuration.

Rule	Iface	Src IP	Dst IP	Protocol	Src Port	Dst Port	State	Action
A	*	*	*	*	*	*	Est.	Accept
B	inet	external	webserver	TCP	> 1023	80	New	Accept
C	lan	internal	external	TCP	> 1023	80	New	Accept
D	lan	internal	external	UDP	> 1023	53	New	Accept
E	*	*	*	*	*	*	*	Drop

Figure 3: Stateful Firewall Configuration

Stateless Firewalls

Stateless firewalls do not generate states for incoming connections but only operate on single packets since keeping states is expensive and needs fast memory. For this reason lookup times are in $\mathcal{O}(\#rules)$ which, for large $\#rules$ is slower than stateful filtering ($\mathcal{O}(1)$). For small $\#rules$ stateless filtering is faster though due to the lack of memory writes. It is also more complex to configure which makes the approach more error prone. An example for a stateless ruleset is shown in Figure 4.

Rule	Iface	Src IP	Dst IP	Protocol	Src Port	Dst Port	Ack	Action
B_1	inet	external	webserver	TCP	> 1023	80	*	Accept
B_2	lan	webserver	external	TCP	80	> 1023	Yes	Accept
C_1	lan	internal	external	TCP	> 1023	80	*	Accept
C_2	inet	external	internal	TCP	80	> 1023	Yes	Accept
D_1	lan	internal	external	UDP	> 1023	53	-	Accept
D_2	inet	external	internal	UDP	53	> 1023	-	Accept
E	*	*	*	*	*	*	*	Drop

Figure 4: Stateless Firewall Configuration

Stateless firewalls usually look at the ACK flag of TCP connection to determine approximately if connections are new or established. When a SYN/ACK packet is sent as first packet of a connection, the firewall will pass it but the host will drop it if implement properly.

Spoofing Protection

Spoofing is the forgery of IP addresses by filling the source IP field with another IP than one's own. Spoofing protection then allows only IP addresses that belong to you on outgoing connections. For incoming traffic this is more difficult since we can not determine where the packet actually came from so we are only able to block our and special purpose IPs.

Common Errors

- How is your firewall management interface reachable?
- What is allowed over the Internet?
- IPv4 and IPv6?
- Outbound rule ANY? (c.f. spoofing)
- Policy's vs. Firewalls understanding of Inbound and Outbound?
- Shadowing: unreachable firewall rules

What Firewalls cannot do

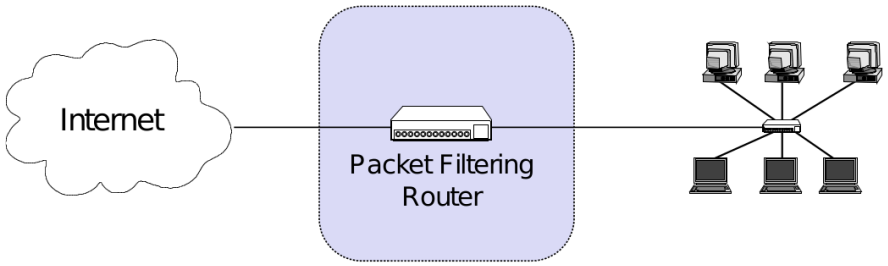
- can't protect against malicious insiders
- can't protect against connections that don't go through it
- can't protect against completely new threats
- can't fully protect against viruses
- does not perform cryptographic operations, e.g. message authentication
- can't set itself up correctly

Bastion Hosts

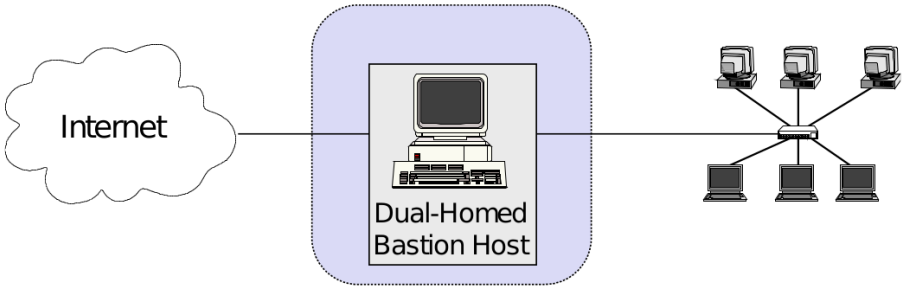
A bastion host is a host that is more exposed to the hosts of an external network than the other hosts of the network it protects. When configuring, one should keep in mind that it might get compromised so do things like disabling SSH password login, do not allow it to sniff internal traffic or disable user accounts.

Firewall Architectures

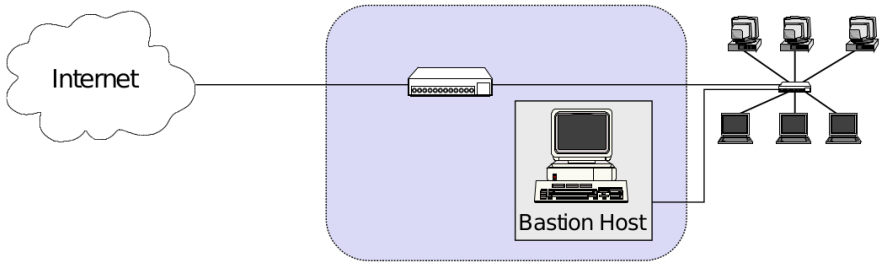
Simple Packet Filter Architecture Packet filtering is done through a filtering router or firewall with two interfaces.



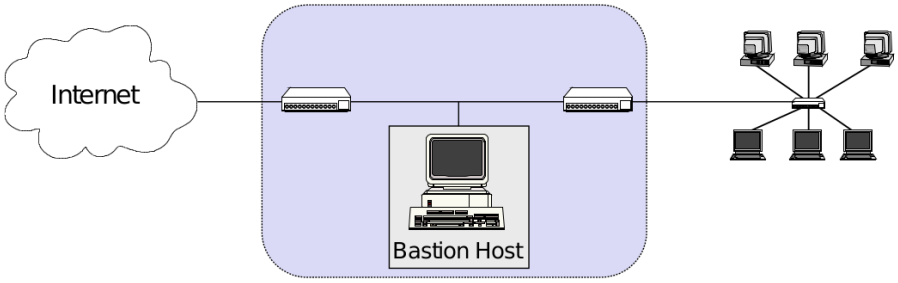
Dual-Homed Host Architecture The bastion host is part of two networks and is firewall and application proxy. Disadvantage: bastion host is bottleneck



Screened Host Architecture The bastion host is proxy, located in the internal network and thus protected by the firewall.



Screened Subnet Architecture - DMZ A demilitarized zone (DMZ) is configured hosting the bastion host (proxy) and publicly accessible servers. The second packet filter is an additional protection measurement in case the DMZ gets compromised.



4 TCP SYN Cookies

4.1 TCP SYN Flood Attack

During an TCP SYN Flood Attack an attacker sends large amounts of packets with spoofed source addresses to the victim. This fills up their connection table with half open connections (sequence numbers) which results in legitimate users not being able to establish new TCP connections. A solution for this are TCP SYN cookies.

4.2 TCP SYN Cookies

TCP SYN Cookies are particularly chosen initial sequence numbers $\alpha = h(S, K_{SYN})$ where K is a secret key, S_{SYN} the source address of the SYN packet and h a cryptographic hash function. On arrival of the ACK message, Bob calculates α again and checks if the ACK number is correct $(\alpha + 1)$. This process is shown in Figure 5.

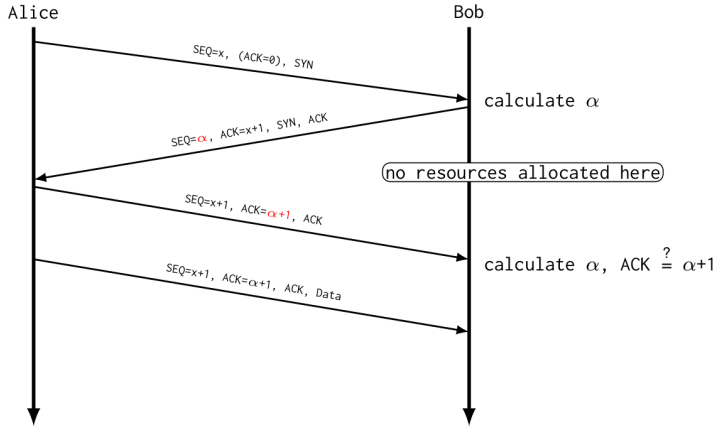


Figure 5: TCP SYN Cookies

Pros

- No resource allocation after SYN packets
- Client does not have to be aware of the server using SYN cookies
- No changes in the TCP protocol necessary

Cons

- Calculating α may be CPU consuming (Linux implementation: CPU local with high cashing efficiency)
- TCP options (like large windows size) cannot be negotiated (Linux implementation: window size hacked into cookie, SYN cookies only enabled if a threshold of connections is exceeded)
- Efficient implementations might be vulnerable to cryptoanalysis (Linux implementation: SHA used, counter updated every minute)

5 Cryptography

Cryptography is the field of science which is concerned with secure communication. It provides cryptographic schemes to achieve confidentiality (ciphers) and authenticity (message authentication codes).

5.1 Security of Cryptographic Schemes

A secure scheme is defined as a scheme, for which it is impossible for a probabilistic polynomial adversary to break it with at most negligible probability. Negligibility is defined as $f(n) < \frac{1}{p(n)}$ for a polynomial p and sufficiently large n, e.g. $f(n) = \frac{1}{2^n}$.
Kerckhoff's principle states that the cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience. Thus only the key needs to be secret.

It is advisable to use only standardized schemes from libraries and RTFM (read the fucking manual) because implementing own schemes has a lot of pitfalls and is more often or almost always done wrong. One also has to keep in mind, that encryption alone does not imply that the system is secure, often integrity and authenticity are more important than confidentiality. And also do not forget key management.

Security of Ciphers

We define a cipher as secure if it is impossible for attackers to recover they key, the entire or even any character of the plaintext from the ciphertext This is the case if it is indistinguishable from a random string with uniform distribution of the character probabilities.

5.2 Hash Functions

Hash functions are functions which take inputs of arbitrary and transform them to a fixed length output. Cryptographic hash functions must also have the following properties:

- **Pre-image resistance**: It is infeasible to find an x' s.t. $H(x') = H(x)$ for given $H(x)$ and randomly chosen x.
- **Second pre-image resistance**: It is infeasible to find an $x' \neq x$ s.t. $H(x') = H(x)$ for a given x.
- **Collision resistance**: It is infeasible to find $x' \neq x$ s.t. $H(x) = H(x')$.

This implies that the that cryptographic hash functions have to be one-way functions. These are functions where it is computationally infeasible to calculate the inverse function.

Cryptographic has functions can be used for authentication, pseudo-random number generation ($b_0 = seed$ and $b_i + 1 = H(b_i|seed)$) or encryption (e.g. in OFB). An example is SHA-3.

5.3 Randomness

We define randomness as unpredictability or entropy. A measure for entropy is the Shannon information entropy calculation

$$H(X) = -\sum_x P(X = x) ln_2(P(X = x))$$

where X is a random variable outputting a sequence of n bits. This entropy and thus randomness is maximized ($H(X) = n$)if bits are uniformly distributed.

Randomness is an important part of cryptography, e.g. for generating keys. In computing, there is no true randomness though since computers are deterministic. For this reason, cryptographically secure pseudo random number generators (CSPRNG) are used which are deterministic algorithms that take truly random, binary input sequences (seed) and output a random-looking sequence of numbers. These unpredictable inputs might either be values resulting from physical phenomena (time between emission of particles, thermal noise of semiconductor, ...) or from software (elapsed time between keystrokes, packet interarrival times, ...).

5.4 Symmetric Cryptography

In symmetric encryption two communication partners Alice and Bob share a secret key that is used for encryption and decryption or message authentication.
We use the following terminology:

- $\leftarrow$ non-deterministic assignment
- $:=$ deterministic assignment
- Key $k \leftarrow Gen(1^n)$
- Plaintext $m = Dec_k(c)$
- Ciphertext $c = Enc_k(m)$
- $Dec_k(Enc_k(m)) = m$

5.4.1 One-Time-Pad (OTP): A Perfect Cypher

For the OTP a perfectly random bitstream opt with the length of the message to encode is needed. Then the encryption operation is defined as $Enc_{otp}(m) = m \oplus otp$ and the decryption operation as $Dec_{otp}(c) = c \oplus otp$
For the OTP to be perfectly secure, the key must only be used once which is impractical in the real world where we usually want $length(k) \ll length(m)$ and thus reusable keys.

5.4.2 Attacking Symmetric Ciphers

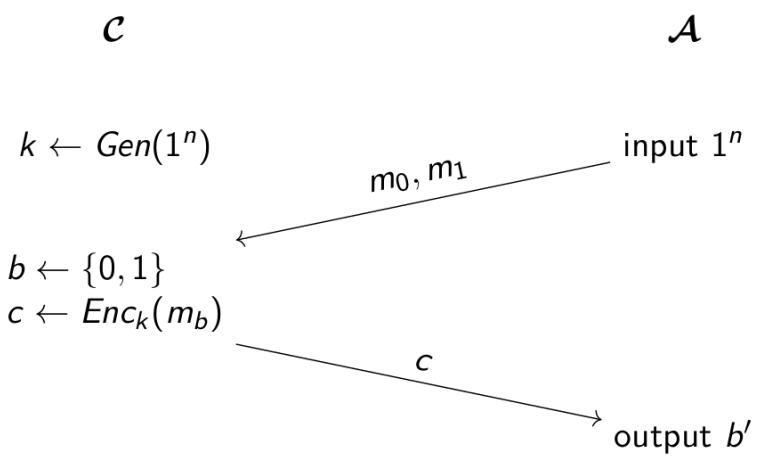
The goals of attacks on ciphers is to learn something about m given a c. Getting any information about k from an attack is also considered a successful one.

To analyze the security, cryptography uses attack games where $\mathcal{C}$ is the challenger and $\mathcal{A}$ the adversary.
Possible attack scenarios are listed in the following, of which some are explained more precisely below:

- Ciphertext-only-attack: attacker knows c
- Known-plaintext-attack: For a fixed k, the attacker got a pair (m, c) and tries to learn something about other ciphertexts
- Chosen-plaintext (CPA) and chosen-ciphertext attack: similar to previous attack, but attacker can chose m or c freely

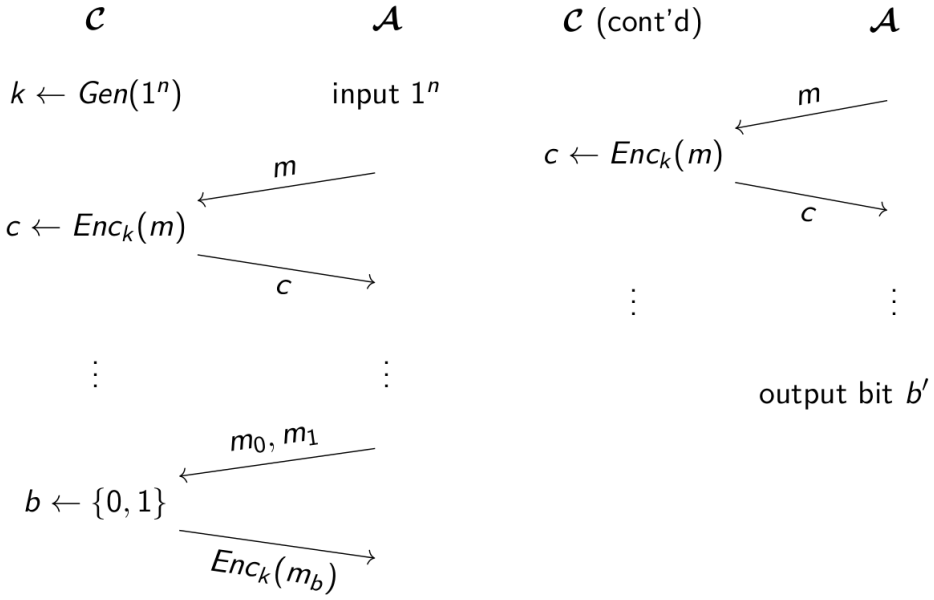
We consider symmetric ciphers to be secure if they withstand CPA.

The Eavesdropping Experiment



In the eavesdropping experiment, $\mathcal{C}$ first generates a random key k. After that, $\mathcal{A}$ sends them two messages m_0 and m_1 for encryption with $|m_0| = |m_1|$. $\mathcal{C}$ then chooses one of the two messages and encrypts it. If $\mathcal{A}$ can tell which message $\mathcal{C}$ chose, the attack was successful. If this happens with probability $0.5 + negligible$ the used cipher is secure.

Chosen-plaintext Attack



In the chosen-plaintext attack the fist step again is that $\mathcal{C}$ generates a random key k. $\mathcal{A}$ is then allowed to send an arbitrary amount of messages to $\mathcal{C}$ who will encrypt and send them back to $\mathcal{A}$ which means that $\mathcal{C}$ is an "oracle" for $\mathcal{A}$. At some point, $\mathcal{A}$ will send two messages m_0 and m_1 of equal length of which $\mathcal{C}$ chooses

one, encrypts it and sends the result back to $\mathcal{A}$. $\mathcal{A}$ is then again allowed to use $\mathcal{C}$ as oracle for an arbitray amount of messages and if they are at some point able to tell which message (m_0 or m_1) $\mathcal{C}$ encrypted, the attack is successful.

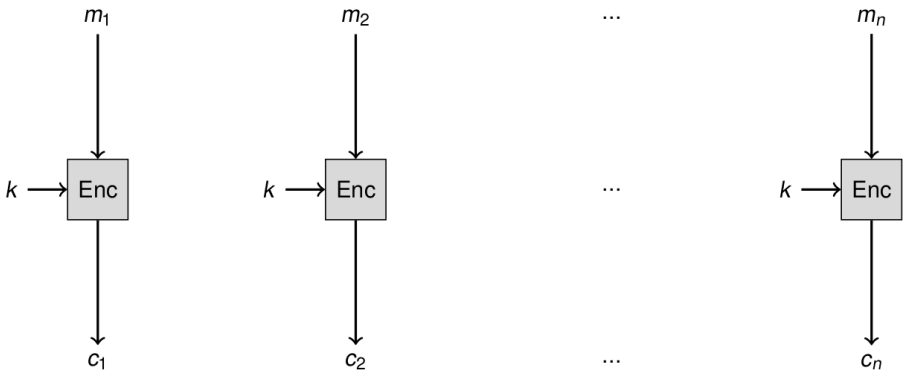
5.4.3 Block and Stream Ciphers

A block cipher encrypts and decrypts inputs of length n to outputs of length n ($\Rightarrow$ block length n). A steam cipher on the other hand generates a random bitstream with arbitrary length, called keystream, that is xored to the plain text to encrypt and decrypt.
The most advisable cipher to use is probably the AES block cipher. It is well tested and proven to be (mostly) secure and hardware supported what makes it quite fast ($> 2GB/s$ with HW support, $200Mbit/s$ without).

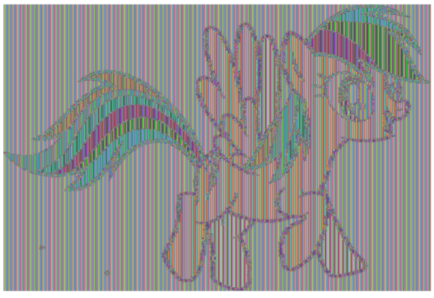
5.4.4 Modes of Encryption

Modes of encryption are necessary to handle messages of variable length with block ciphers. The plaintext therefore is split into parts of length equal to the cipher block length. If the last block is shorter than the block length, we add padding.

5.4.5 Electronic Code Book Mode (ECB)



In ECB, every plaintext block is encrypted with the unmodified key as input. The problem thereby is that identical plaintext blocks result in the same ciphertext blocks.



5.4.6 Cipher Block Chaining Mode (CBC)

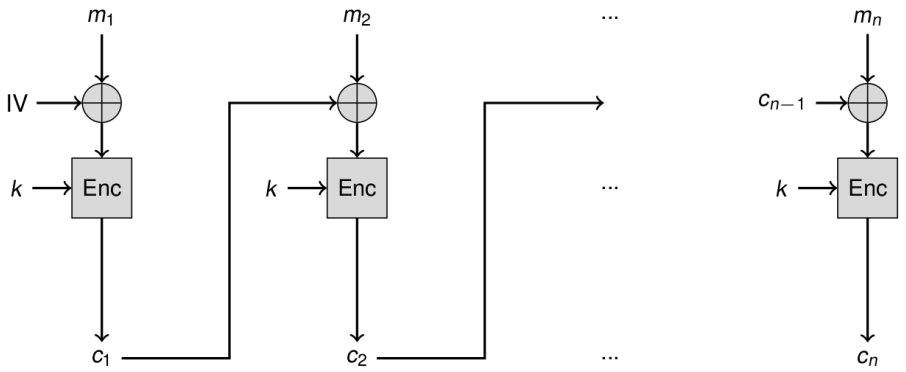


Figure 6: CBC Encrypt

CBC uses the ciphertext from the previous block xored with the current message block and the unmodified key k as inputs of the encryption function $c_i = Enc_k(c_{i-1} \oplus m_i)$. For the first message, where no previous cipher block is available, an initialization vector (IV) is used which may be sent in plaintext and is fresh for every massage (or packet if the message is split across multiple packets).
Compared to ECB, the advantage is that equal plaintext blocks/messages are not encrypted to the same ciphertext blocks/messages.

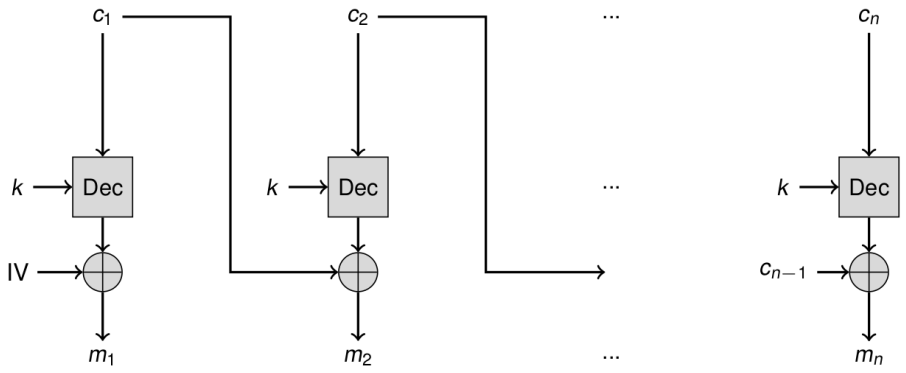


Figure 7: CBC Decrypt

Decryption is defined by $m_i = c_{i-1} \oplus Dec_k(c_i)$.

5.4.7 Output Feedback Mode (OFB)

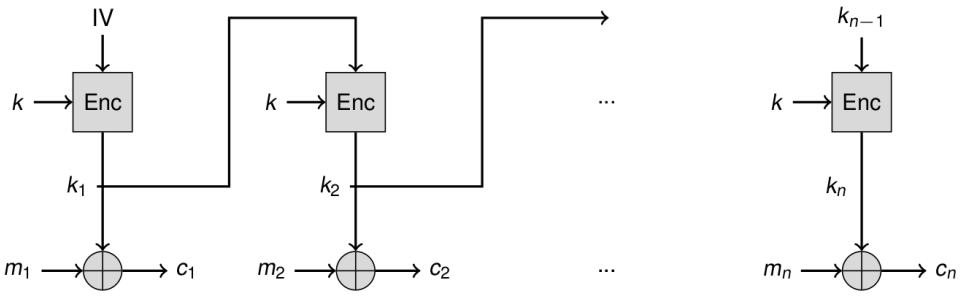


Figure 8: OFB Encrypt

OFB transforms a block cipher into a stream cipher. In step i, it takes cipher key k_{i-1} form the previous step and encrypts it with the key k. This cipher key k_i is then xored with the corresponding plaintext block m_i to calculate the ciphertext c_i. In the first step an IV is used as k_{i-1}.

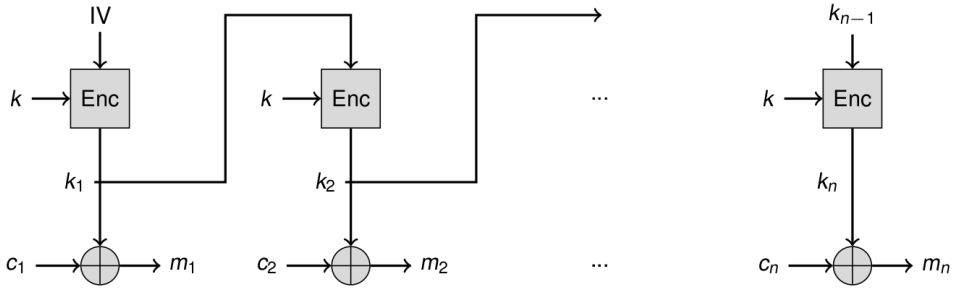


Figure 9: OFB Decrypt

Decryption is the same procedure except that the key stream is xored with the cipher text blocks.

5.4.8 Counter Mode (CTR)

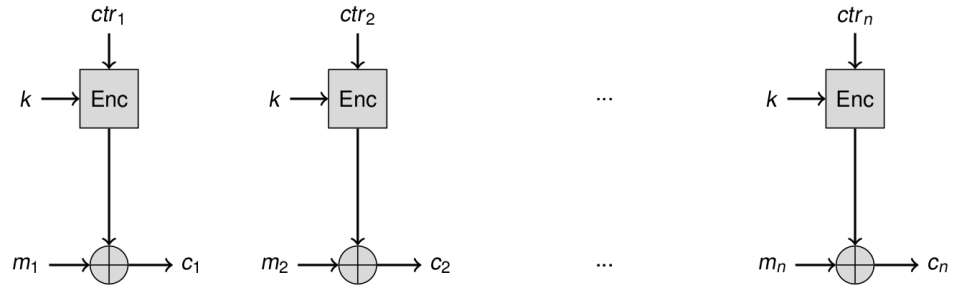


Figure 10: CTR Encrypt

In step i, CTR encrypts a counter $ctr_i = IV||i$ with key k and xors the result with the plaintext block m_i. Like OFB, CTR also transforms block ciphers into stream ciphers.

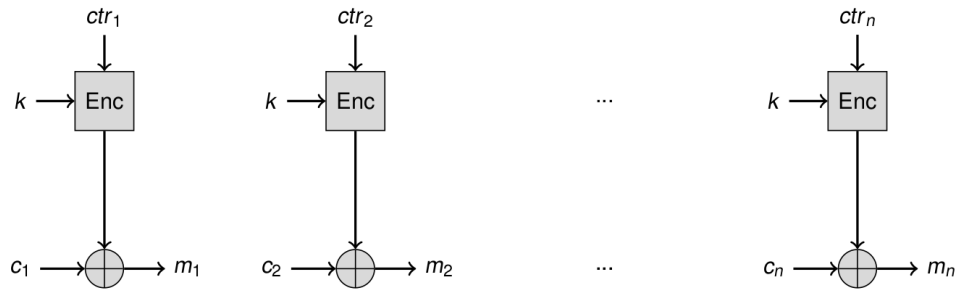


Figure 11: CTR Decrypt

Decryption is done by $m_i = Enc_k(ctr_i) \oplus c_i$.

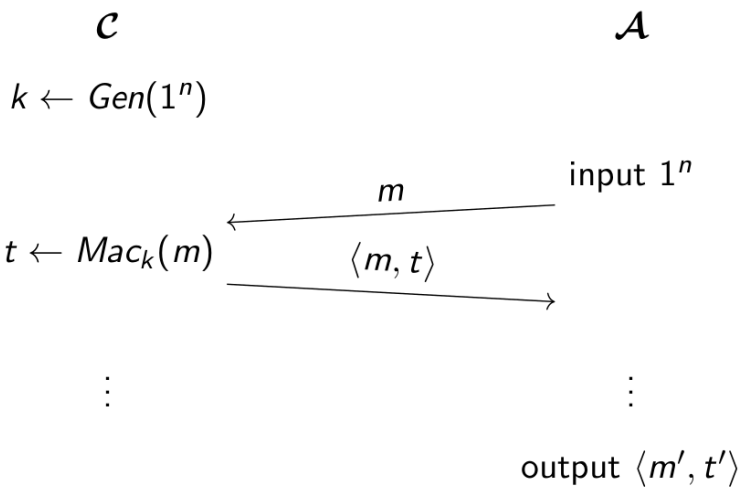
5.4.9 Message Authentication Code (MAC)

MACs are used in symmetric cryptography to ensure message authenticity but do not provide replay protection. They are transmitted with the message $\langle m, t \rangle$. In practice, MACs are either based on hash functions (HMAC), CBC or included in the encryption (authenticated encryption modes).
We define the following terminology:

- Key $k \leftarrow Gen(1^n)$
- Tag $t \leftarrow Mac_k(m)$
- Verification $b := Vrfy_k(m, t)$ where $b = 1$ means valid and $b = 0$ invalid

We define a MAC as secure, if it withstands the adaptive chosen-message attack.

Adaptive Chosen-message Attack



In the adaptive chosen-message attack $\mathcal{C}$ first generates a random key k. $\mathcal{A}$ then sends an arbitrary amount of messages to $\mathcal{C}$ for which they answer with a message consisting of the original message and a tag t. At

some point, $\mathcal{A}$ will generate a message-tag pair $\langle m', t' \rangle$ where m was not previously sent to $\mathcal{C}$ The attack is successful, iff $Vrfy_k(m', t') = 1$, so if $\mathcal{A}$ was able to generate a valid MAC tag for a new message.

Hash MAC (HMAC)

HMAC uses cryptographic hash functions to calculate the MAC. It is calculated with

$$t = H(K \oplus opad \mid H(K \oplus ipad|m))$$

To note here is that K is first extended to the block length required for the input of the hash function by appending zeros. *opad* and *ipad* are arbitrary values with the restriction that they have a large Hemming distance to each other.
Another note: $H(K \mid m)$, $H(m \mid K)$ or $H(K \mid p \mid m \mid K)$ are not secure.

CBC-MAC

CBC-MAC encrypts the message to authenticate with another key than used for encryption and uses the last ciphertext block as MAC. CBC is shown in Figure 6. If the length of a message is not known or no other protection exists, CBC-MAC can be prone to length extension attacks. CMAC resolves this.

CMAC

CMAC is an improved version of CBC-MAC which has to keys, calculated from the shared key, where one is used for xoring with complete bocks before encryption and the other one for incomplete blocks. XCBC-MAC is another improvement of this where the two keys are input to the algorithm and not derived from k.

5.4.10 Combining Confidentiality and Authentication

Generally the best approach to combine confidentiality and authentication is to first encrypt and then authenticate, i.e. $c \leftarrow Enc_{k_1}(m)$, $t \leftarrow Mac_{k_2}(c)$ and transmit $\langle c, t \rangle$.

5.5 Asymmetric Cryptography

In asymmetric cryptography, the two communication partners no longer have one shared key, but each participant has a key pair. A private/secure key (sk) that is only known to one person and is used for decryption and message signing and a public key (pk) that can be published and is used for encryption and message verification. Asymmetric cryptography is based on mathematical problems which are believed to be hard (e.g. integer factorization). It only needs an authenticated channel to exchange keys of which less amounts are required than in the symmetric setting. It is orders of magnitudes slower though.

5.5.1 Public-key Encryption Scheme

1. $(pk, sk) \leftarrow Gen(1^n)$
2. Encryption with public key: $c \leftarrow Enc_{pk}(m)$
3. Decryption with private key: $m := Dec_{sk}(c)$

RSA

Key generation in RSA is done with the following procedure:

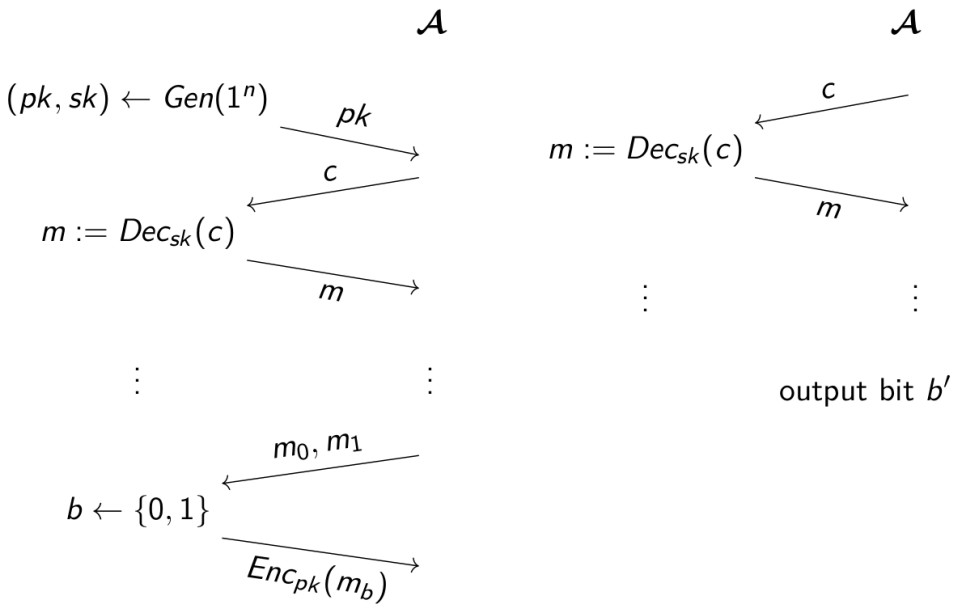
1. $(N, p, q) \leftarrow GenModulus(1^n)$
2. $\phi(N) := (p-1)(q-1)$

3. find e: $\gcd(e, \phi(N)) = 1$
4. $d := [e^{-1} \mod \phi(N)]$
5. $pk = \langle N, e \rangle$
6. $sk = \langle N, d \rangle$

Encryption then is defined as $RSA_{pk}(y) := [y^e \mod N]$ and decryption as $RSA_{sk}(y) = [z^d \mod N]$.

Chosen-ciphertext Attack (CCA)

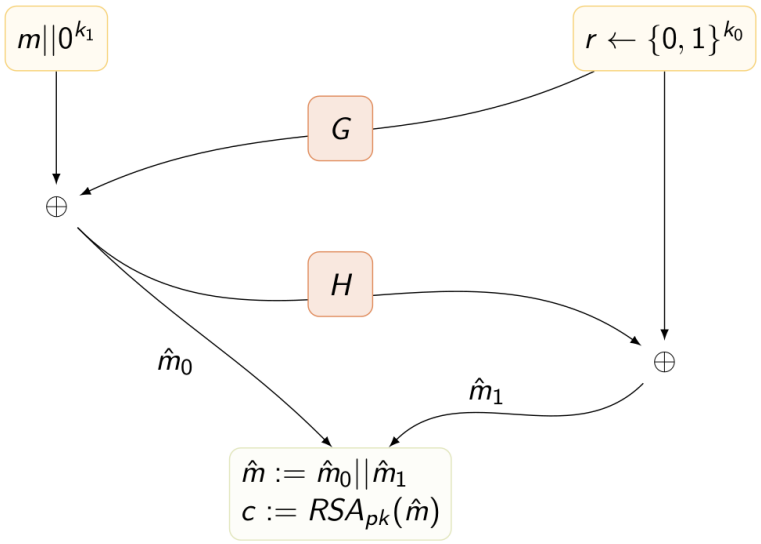
An asymmetric encryption scheme is considered secure if it withstands CCA.



CCA is quite similar to CPA except $\mathcal{A}$ sends ciphertext messages to $\mathcal{C}$ for probing instead of plaintext ones. A restriction not depicted in the illustration above is that $\mathcal{A}$ may not request a decryption for $Enc_{pk}(m_b)$ itself.

Optimal Asymmetric Encryption Padding (OAEP)

OAEP is a padding scheme for asymmetric encryption depicted below.



RSA used in conjunction with OAEP is secure under CCA.

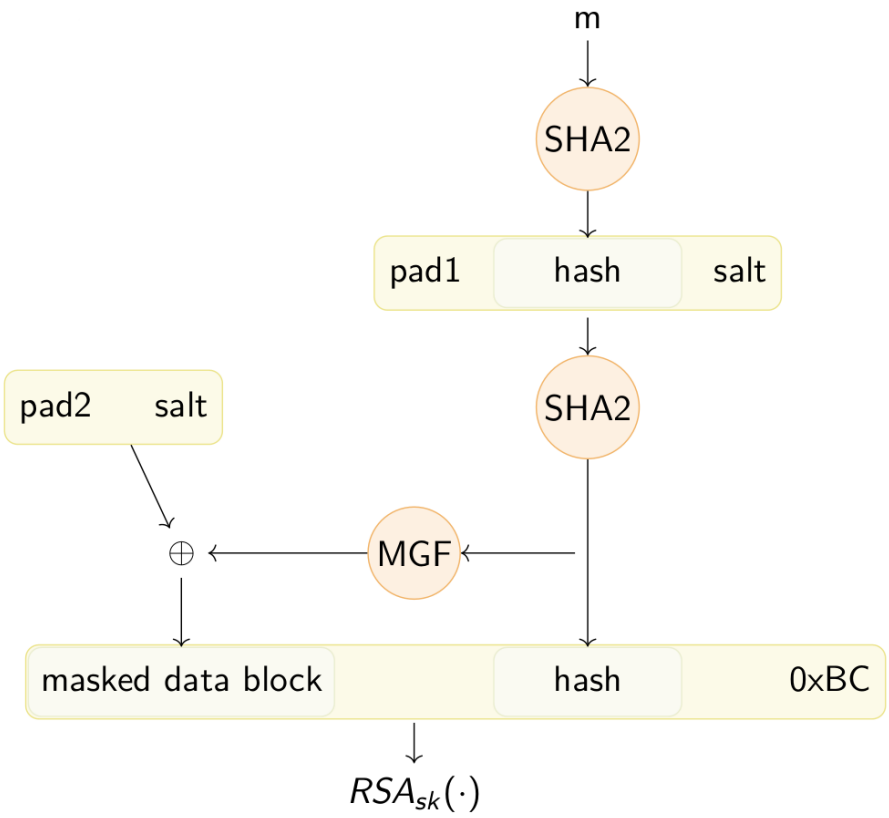
5.5.2 Signature Scheme

1. $(pk, sk) \leftarrow Gen(1^n)$
2. Signature generation with private key: $\sigma \leftarrow Sign_{sk}(m)$
3. Signature verification with public key: $b := Vrfy_{pk}(m, \sigma)$ where $b = 1$ means valid and $b = 0$ invalid

Signatures are often slower than their MAC relatives and provide non-repudiation. A signature scheme is considered secure if it withstands the adaptive chosen-message attack.

RSASSA-PSS

RSASSA-PSS is a signature scheme that uses RSA.



RSASSA-PSS is secure under adaptive chosen-message attack and breaking it would mean that RSA is broken.

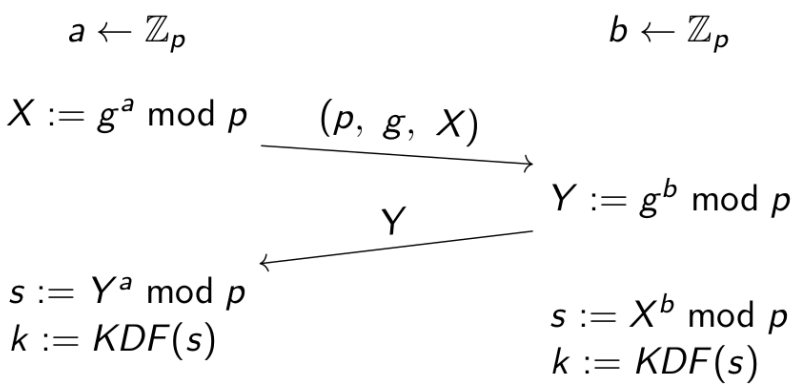
5.6 Hybrid Approach

Since public key cryptography is slower than secret key cryptography, a hybrid approach is often advisable. Public-key cryptography is used to protect a shared key (session key) that is used for any further communication via secret-key cryptography. They private key for the key exchange is called long-term key.

Perfect forward secrecy is often an important property of this approach. It states that if an attacker gains access to the long-term key or the session key, they should not be able to decrypt messages of past sessions.

A common strategy is to use a signed Diffie-Hellman key exchange and a secret-key authenticated encryption scheme for communication. To guarantee perfect forward secrecy, DH keys have to be calculated for every connection and old keys have to be wiped.

5.6.1 Diffie-Hellman Key Exchange (DH)



In the first step of the DH key exchange, communication partner A selects a prime p, a generator g which is a primitive root for the cyclic group of $\mathbb{Z}_p$ and a random number $a \in \mathbb{Z}_p$. A then calculates $X = g^a \mod p$ and sends it with p and g to B. B also chooses a random number $b \in \mathbb{Z}_p$ and calculates $Y = g^b \mod p$ which they send back to A. Both can then calculate the key k by $k = Y^a = g^{ba} = g^{ab} = X^b \mod p$.

6 Secure Channel

A secure channel is a confidential, integrity protected and authenticated communication between two parties where messages are received in the correct order, no duplicates occur and if messages are missing we know which ones.

We will use the previously discussed cryptographic schemes to achieve this. Despite knowing our modules to achieve most (confidentiality, authenticity) of the points from above, there are still some freedoms to put them together.
MAC-then-Encrypt $(Enc_{k-enc}(m, MAC_{k-int}(m)))$first authenticates the message and then encrypts the message as well as the MAC.
SSL uses this scheme, but many attacks on the protocol result from this. MAC-and-Encrypt $(Enc_{k-enc}(m), MAC_{k-int}(m))$ calculates the MAC and simply appends it to the encrypted message. This approach is considered the weakest.
Encrypt-then-MAC $(Enc_{k-enc}(m), MAC_{k-int}(Enc_{k-enc}(m)))$ is the reversed approach compared to MAC-then-Encrypt. It first encrypts the message and then calculates the MAC based on the encrypted message. This approach is considered to be the most secure.
For loss and order detection it is possible to simply use a counter appended to the initial message. It is necessary that it is at least authenticated. Another note to be made here is that one has to make sure on implementation level that timing attacks are not possible. Timing attacks look at how long certain operations take to get information about the system.

6.1 Authenticated Encryption with Associated Data (AEAD)

AEAD stands for a secure channel where data is sent that is first encrypted and then authenticated (Encrypt-then-MAC). Additional to that, other data may be sent unencrypted but also authenticated. Such data might be IVs, message counters, information necessary for message routing among others.
Special algorithms exist which need only one pass over the data to do this, most take two though.

6.2 Offset Codebook Mode (OCB)

OCB is such an one run encrypt and authenticate method.

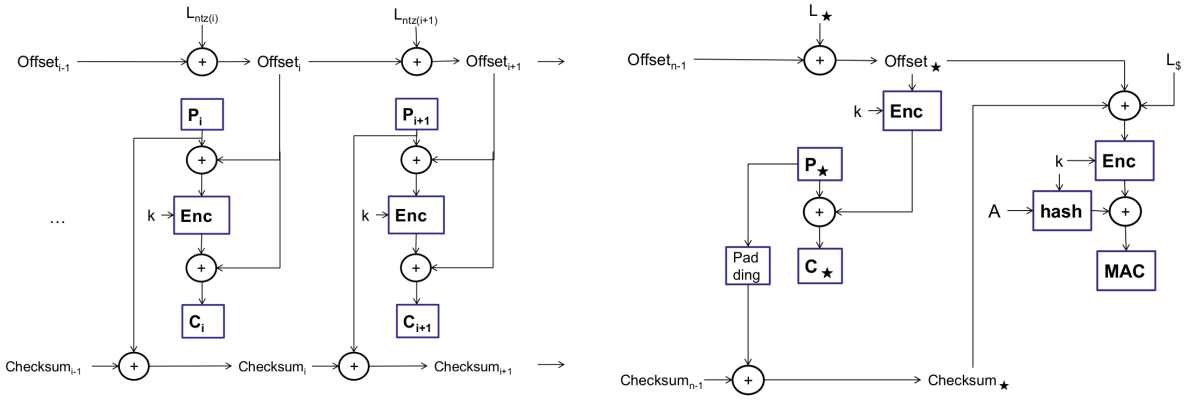


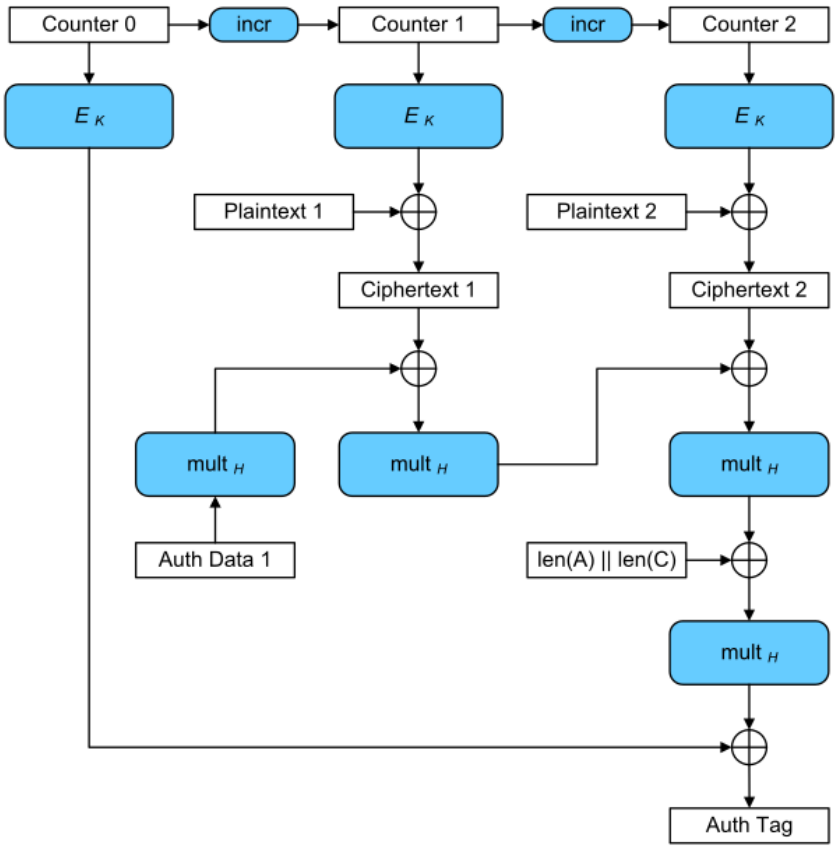
Figure 12: OCB

OCB works as described above with $L_* = Enc_k(0)$, $L_\$ = double(L_*)$, $L_0 = double(L_\$)$, $L_i = double(L_i) = double(L_{i-1})$ and ntz = number of trailing zeros. Most of the times only a view values for L_i are used, but

they can be precomputed and stored in a lookup table. To note here is that OCB only needs one key for encryption and authentication but a fresh nonce for each run.

6.3 Galois/Counter Mode (GCM)

GCM is another one run approach to encrypt and authenticate. It combines the concept of the counter mode for encryption with Galois Field Multiplication (based on $x^{128} + x^7 + x^2 + x + 1$) to compute the MAC on the ciphertext.



GCM generates check values for *Auth Data*, which is data that is not to be encrypted, by xoring and GF multiplication with $H = Enc(k, 0)$.

6.4 Attacks against a Secure Channel

6.4.1 Stream Ciphers

Steam ciphers can be vulnerable to attacks if using the same IV multiple times. Consider the following, where $KS(IV, k)$ is the keystream generated with key k and initialization vector IV and C_i is the ciphertext calculated from the plaintext P_i and the keystream

$$C_1 = P_1 \oplus KS(IV, k)$$
$$C_2 = P_2 \oplus KS(IV, k).$$

Xoring C_1 and C_2 results in

$$C_1 \oplus C_2 = P_1 \oplus KS(IV, k) \oplus P_2 \oplus KS(IV, k) = P_1 \oplus P_2$$

So the attacker can recover $P_1 \oplus P_2$ from C_1 and C_2 and if they know P_1 or P_2 they can recover the other.

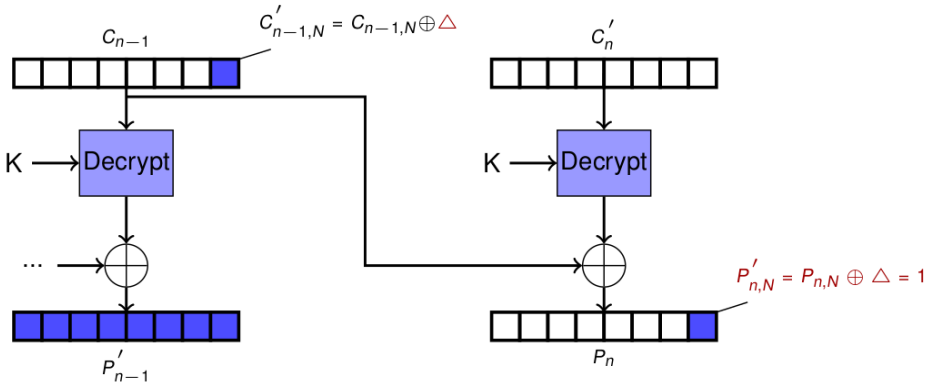
6.5 Padding Oracle Attack

The padding oracle attack can reduce the attack time from $\prime(N^L)$ to an average of $L \cdot \frac{N}{2}$ tries (N is the size of the alphabet, L the password/key length) under certain assumptions by decrypting ciphertexts bitwise.

1. Attacker gets a ciphertext C with n blocks of length N
2. Oracle that sends padding errors for wrong paddings and MAC errors if padding is correct. Can also be done with side channels like timings, ...
3. MAC-then-Encrypt is used (more precisely MAC-then-Encode-then-Encrypt, where first the MAC is calculated from the plaintext, then everything is padded to the block size and then encrypted)
4. CBC is used
5. PKCS7 padding scheme is used: 1 byte padding $\rightarrow$ 1, 2 byte padding $\rightarrow$ 2 2, ...

The attack starts decrypting the last byte of the last block of the massage $P_{n,N}$ by altering $C_{n-1,N}$ by xoring a value $\triangle$ to it $C'_{n-1,N} = C_{n-1,N} \oplus \triangle$. $C'_{n-1,N}$ then is sent to the oracle. If it responds with a padding error try again with a new $\triangle$ (max 256 tries, 1 byte!). If it returns a MAC error though, we know that padding was fine and thus that $P_{n,N} = 1$ (PKCS7 padding scheme!). With that information we then can calculate $P_{n,N}$.

$$P'_{n,N} = P_{n,N} \oplus \triangle = 1 \rightarrow P_{n,N} = 1 \oplus \triangle$$



Now we want to continue with $P_{n,N-1}$ for which we need a padding of length 2. Since $P_{n,N}$ known, we can calculate $C'_{n-1,N}$ so that $P_{n,N} = 2$ with

$$P_{n,N} \oplus C'_{n-1,N} = 2 \rightarrow C'_{n-1,N} = P_{n,N} \oplus 2$$

To find $P_{n,N-1}$ now, find a $C'_{n-1,N-1}$ that satisfies $C'_{n-1,N} \oplus P_{n,N-1} = 2$ with the same procedure as before. To completely decrypt C_n we have to repeat these steps until all bytes of the block are decrypted. To decrypt C_{n-1} we can just simply cut off C_n and do the same again. For C_1 the IV is used as ciphertext for the attack modifications.

7 Public Key Infrastructures (PKI)

A PKI is an infrastructure that handles certificates in a untrusted network. Certificates are cryptographic bindings between an identifier and a public key that is to be associated with that identifier. The identifier often refers to persons or businesses, but can also be other attributes like access rights. What is always necessary is that it has to be verified that the identifier and corresponding key belong together. In case of persons or businesses, additionally a verification that the entity behind the name is the entity it claims to be.

PKIs are created by issuing certificates between entities. An certificate issuer, which is a trusted third party and whose certificate and thus public key (K_{I-pub}) we know, creates a signature for a third party X and its public key K_{X-pub} with its private key K_{I-priv}. The tuple of $(X, K_{X-pub}, Sig_{K_{I-priv}(X|K_{X-pub})})$ is then called a certificate. In reality, other attributes are stored in certificates additionally.

PKIs can either be hierarchical or not. In hierarchical PKIs the issuers are often called certificate authorities (CA), in non-hierarchical ones endorsers. The first mentioned type here has one central CA. This is problematic though because who is to decide that the CA is trustworthy (In reality this trust management is done by operating systems or browsers). Furthermore there is a high load on the CA which can lead to sloppy work which results in mistakenly issued certificates. In some cases hierarchical CAs have intermediate registration authorities (RA) that do the verification step of identifying X but do not issue certificates. But this does not resolve the problem that there still is only one CA but merely reduces its workload. The solution for this used in reality is to have multiple CAs which represent different legislations and which are all accepted equally. This may increase the danger that one fucks up though.
In non-hierarchical PKIs every participant can issue certificates. Different trust metrics are used to automatically reason about authenticity of the bindings between entity and key like defining rules for how many levels of indirections are allowed.

7.1 X.509

X.509 is the certificate standard used in SSL/TSL and thus in a large part of the Internet. It relies an root

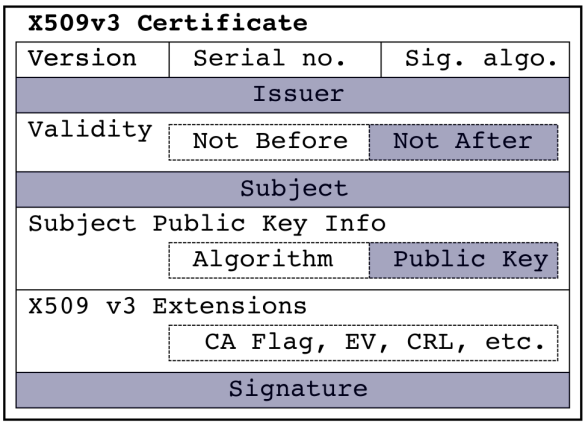


Figure 13: X.509 Certificate

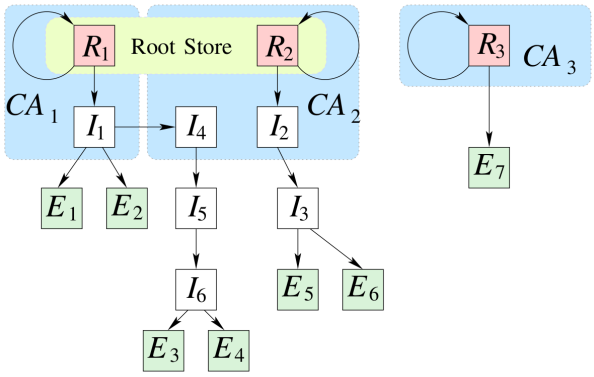


Figure 14: X.509 Root Stores

stores which contain certificates of trusted CAs. These root stores are usually managed by the operating system or the browser. Root store certificates then usually are used to sign intermediate certificates first instead of actual host certificates because this way root certificates can be protected and signing can be delegated to sub-CAs. A special form of this is cross signing where CAs sign a certificate of other CAs. This undermines the trust chain of the Internet though.

7.1.1 Certificate Issuance

Certificates are either validated with domain validation (DV) or an Extended Validation (EV). In DV, one has to send an e-mail to the chosen CA with information about the domain (whois details, DNS txt record, address is well known contact of the domain, ...). The ownership of the e-mail address thereby validates the ownership of the domain. In EV, strong legal documents are required to prove identities.
Also approaches that lay in between are possible.

7.1.2 Certificate Revocation

Certificates have to be valid at the time of interest (e.g. a site is requested)and has therefore to be proven for revocation at that point. This can be done in several ways.
The first approach are **Certificate Revocation Lists (CRL)** which are lists of revoked certificates issued, maintained and updated by CAs. Browsers then should download these lists every time an CRL is updated and check for revoked certificates on site requests. In reality CRLs are barely used since time intervals between CRL updates where the system is vulnerable, the updates can be blocked by MITM and CRL can grow quite large.

Another approach is the **Online Certificate Status Protocol (OCSP)** where CRL-like data structures are maintained on the server-side. In practise, every time a TLS handshake is done (not every request, inperformant), a request is sent to these servers asking if a certificate is revoked (possible answers: good, revoked, unknown). The approach of not sending OCSP requests for every site request but bundling them together (in the TLS handshake) is called OCSP stapling.

Some other approaches for certificate revocation are in-browser revocation lists which maintain revoked certificates for the most important sites or short-lived certs.

7.1.3 Pinning/TOFU

A suggestion to enhance X.509 is to use pinning/trust on first use (TOFU). The idea is that hosts send information about themselves on first contact (e.g. public key) which clients then store for later identification if they decide to trust the host. This pinning can be static, i.e. there are preloaded pins provided by the Chrome or Firefox, or dynamic, i.e. helpful information is communicated to aid clients with pinning.
This concept cannot be applied for certain users though, for example citizens of an authoritarian country. Also servers might update their keys, which might look random to clients which results in them not trusting the host and browsers cannot come with preloaded keys for all sites and keep them up to date.

7.1.4 Public Log Schemes

The idea behind public logs is to store some information about how issues certificates to who. They are neutral, signed and append-only lists of certifications.

In X.509 the log scheme is called Certificate Transparency (CT). Monitors and Auditors permanently watch the log to verify its consistency and the inclusion of new certificate chains. Thereby monitors primary function is to verify the append-only property and anyone is able to run one. Their may also keep copies of logs and search for violating certificate issuances. Auditors are less powerful entities which the log's consistency. This is usually done by browsers.
To perform these checks and proofs, a Merkle tree structure (hash tree) is used to store the logs.
To simplify the inclusion proof of a certificate issuance in a log, signed certificate timestamps (SCT) can be used. They represent vaguely a receipt you get if you enter a certificate into a log. If an auditor gets a SCT of a log it trusts, it (nearly) replaces the traditional inclusion proof.

8 Cryptographic Protocols

A Cryptographic protocol is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods, often as sequences of cryptographic primitives. A protocol describes how the algorithms should be used.

8.1 Attacks on Cryptographic Protocols

- Replay attack: replay messages
- Oracle attack: ask oracles about stuff one can normally not do
- Typing attack: Replace message field of one type with on of another type
- Modification: Attacker alters messages sent.
- Preplay: The attacker takes part in a protocol run prior to a protocol run.
- Reflection: The attacker sends back protocol messages to principles who sent them. Related to Oracle attacks.
- Denial of Service: The attacker hinders legitimate principles to complete the protocol.
- Certificate Manipulation: Attacks using manipulated or wrongly-obtained certificates.
- Protocol Interaction: Make one protocol interact with another, e.g. by utilizing that principles use the same long-term keys in both protocols and utilizing that for an attack.

8.1.1 Desirable Properties of Cryptographic Protocols

- Authenticity
- Integrity
- Confidentiality
- Replay protection
- Forward Secrecy and Key Agreement
- Scalability
- Avoidance of Single-Points-of-Failures
- Selection of Algorithms
- Generic Authentication Methods
- Simplicity
- ...

8.2 Designing an own protocol

In this chapter we will try to build our own cryptographic protocol, including intermediate steps and attacks on those intermediate results. This will show common pitfalls and the general process for/of this.
Our communication partners will be named Alice and Bob.

The first step of our design is the entity authentication and the key establishment. These are usually done together because if keys are established unauthenticated, we can not be sure that the key is actually shared among the intended communication partners. Keys need to be exchanged in order to have a secure channel to communicate, not regarding a physically secure connection or such. In order to establish session keys Alice and Bob either must have a long-term shared key, each others public keys or an exchanged key with a trusted third party (TTP). The latter approach thereby is the most scaleable one in terms of key management. Our first try for this is shown in Figure 15. In the figure the $_K$ means an encrypted and integrity-protected message.

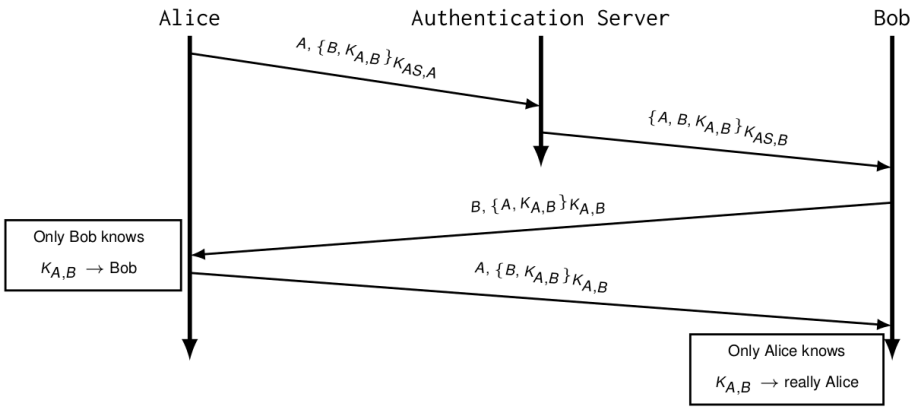


Figure 15: Protocol Try 1

To prevent replay attacks, it is advisable to use fresh nonces for every protocol run. Figure 16 shows a possible attack and Figure 17 an improved protocol.

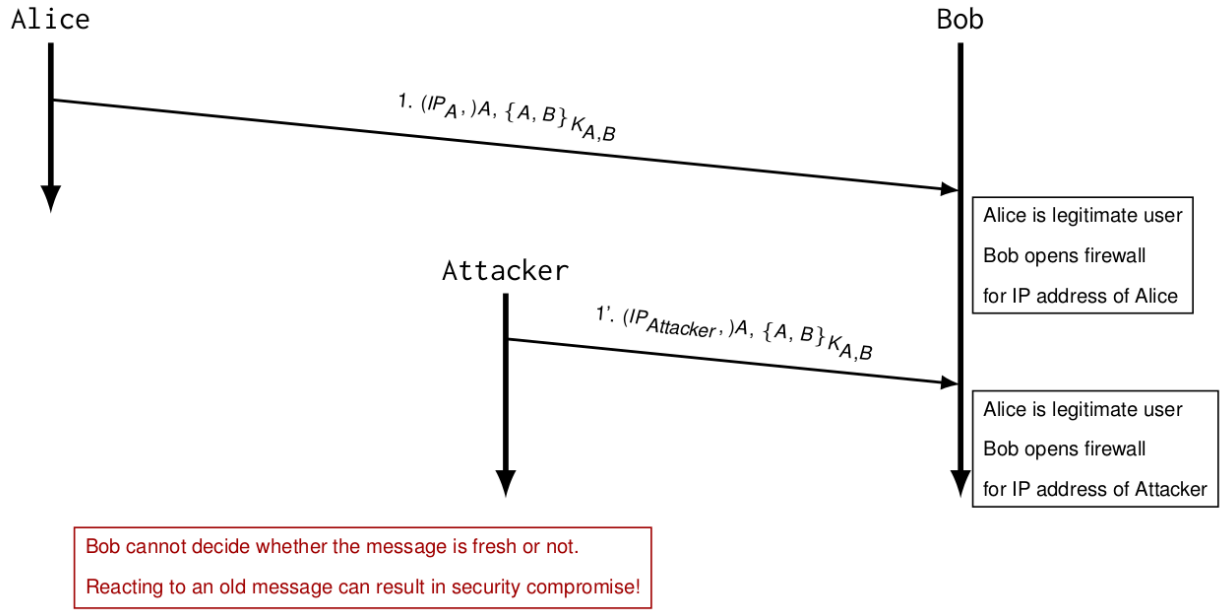


Figure 16: Replay Attack on our Protocol

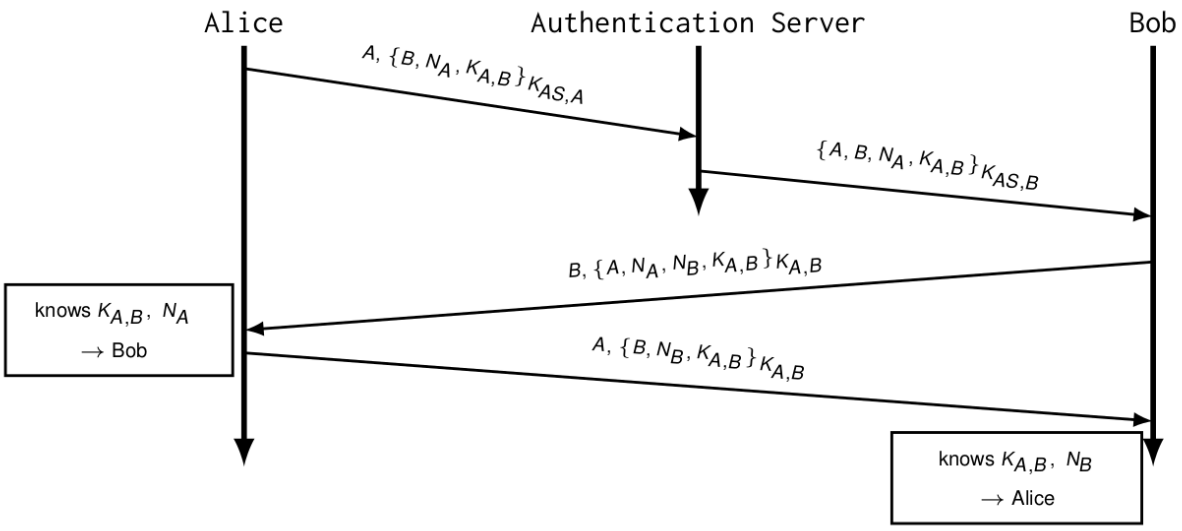


Figure 17: Protocol with Nonces

Having dealt with that, another problem presents itself: oracle attacks are still possible (c.f. Figure 18).

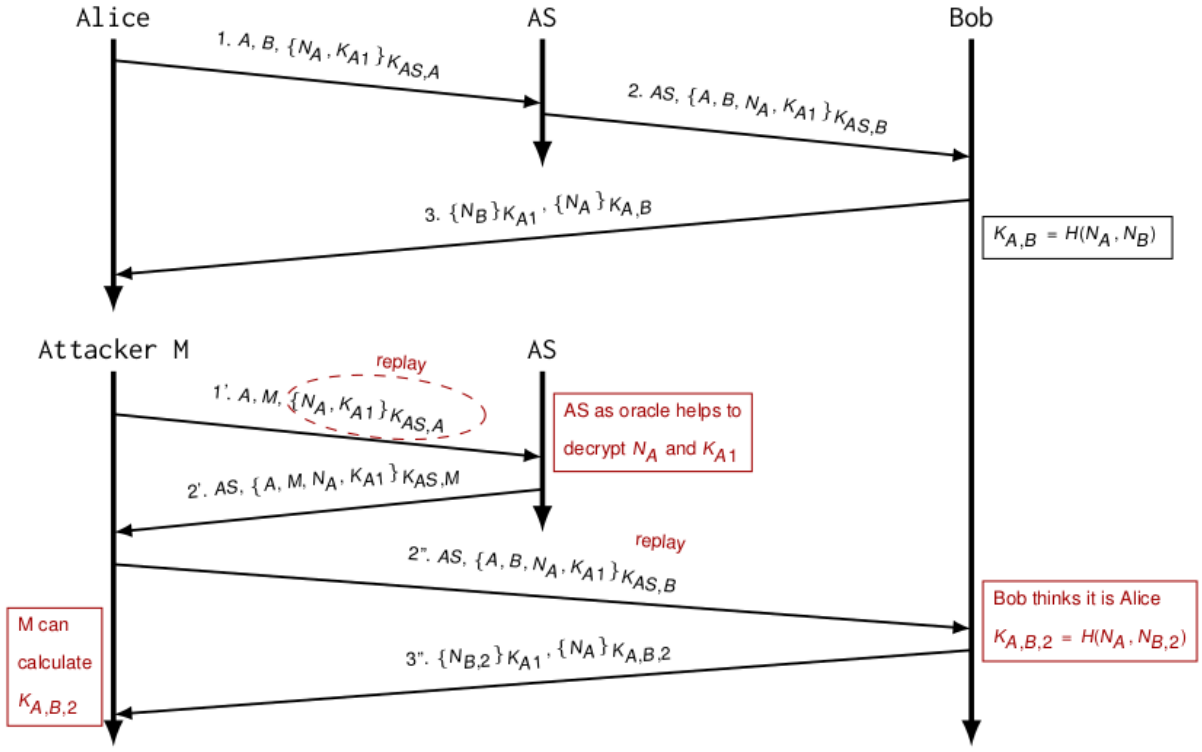


Figure 18: Oracle Attack on our Protocol

Now we can add forward secrecy by using Diffie-Hellman as shown in Figure 20. DH_B is integrity protected because $A, N_A, N_{B\,K_{A,B}}$ is already encrypted with the key derived from the Diffie-Hellman exchange. If DH_B is changed by an attacker, Alice cannot decrypt the message anymore.

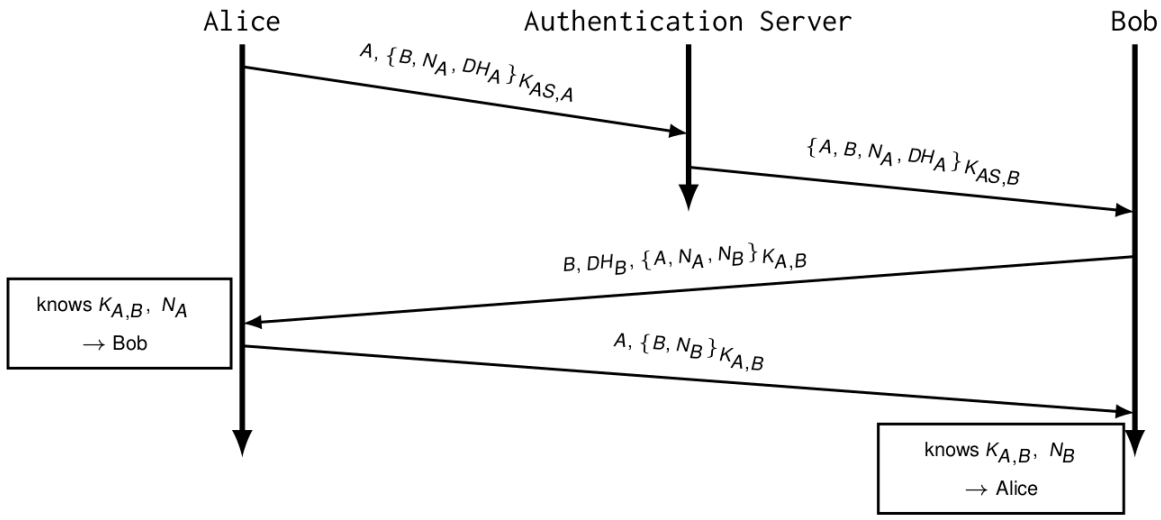


Figure 19: Protocol with Forward Secrecy

To improve reliability, we remove the single point of failure, i.e. the authentication server, from single protocol runs, It might still be used beforehand to provide keys or certificates.

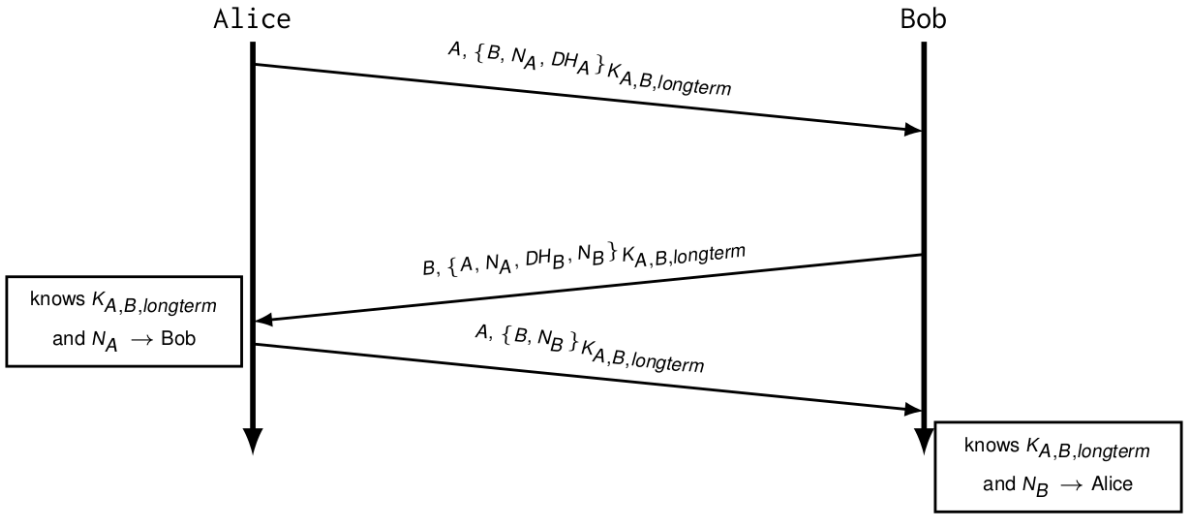


Figure 20: Protocol without Authentication Server

To simplify key management and authentication, we want to use public key cryptography. Furthermore we want to include a mechanism that lets Alice and Bob determine which crypto algorithms to use (e.g. to stay on the state of the art without changing the protocol).

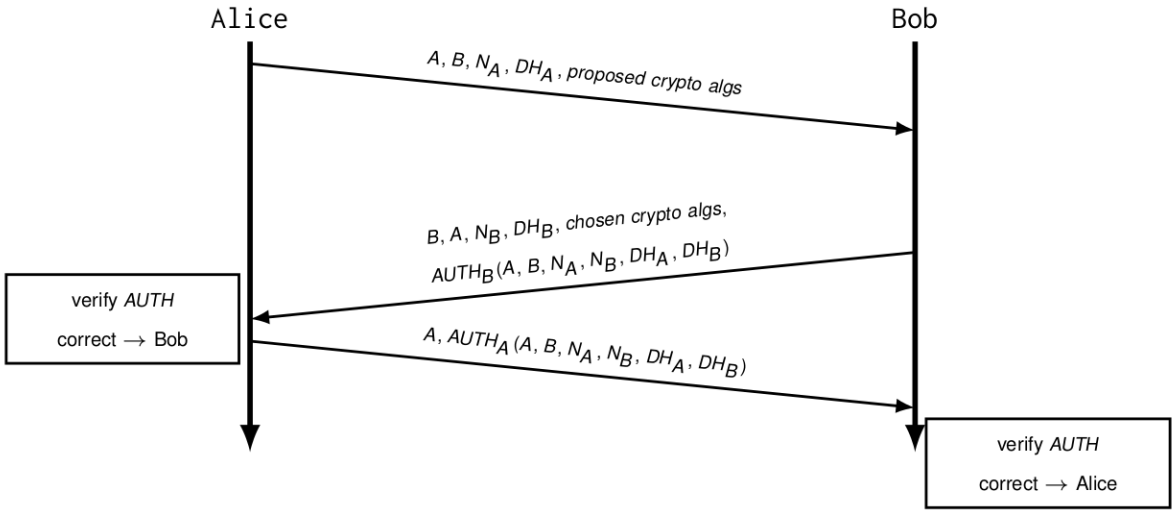


Figure 21: Protocol with Public-key Cryptography and Crypto Algorithm Handshake

In case symmetric cryptography is used for AUTH, a replay attack is possible because Alice and Bob authenticate identical messages with the same key in that scenario. For this reason, we change the authenticated messages.

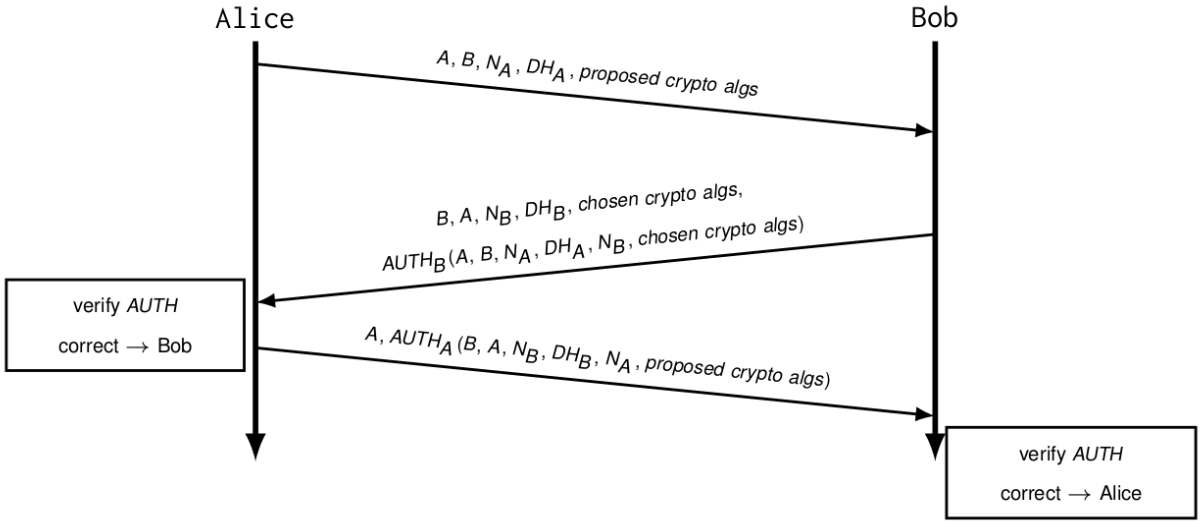


Figure 22: Protocol with different Authenticated Messages

To decrease complexity, what makes analysis harder and the attack surface bigger, we split the second message into two (request-response pairs).

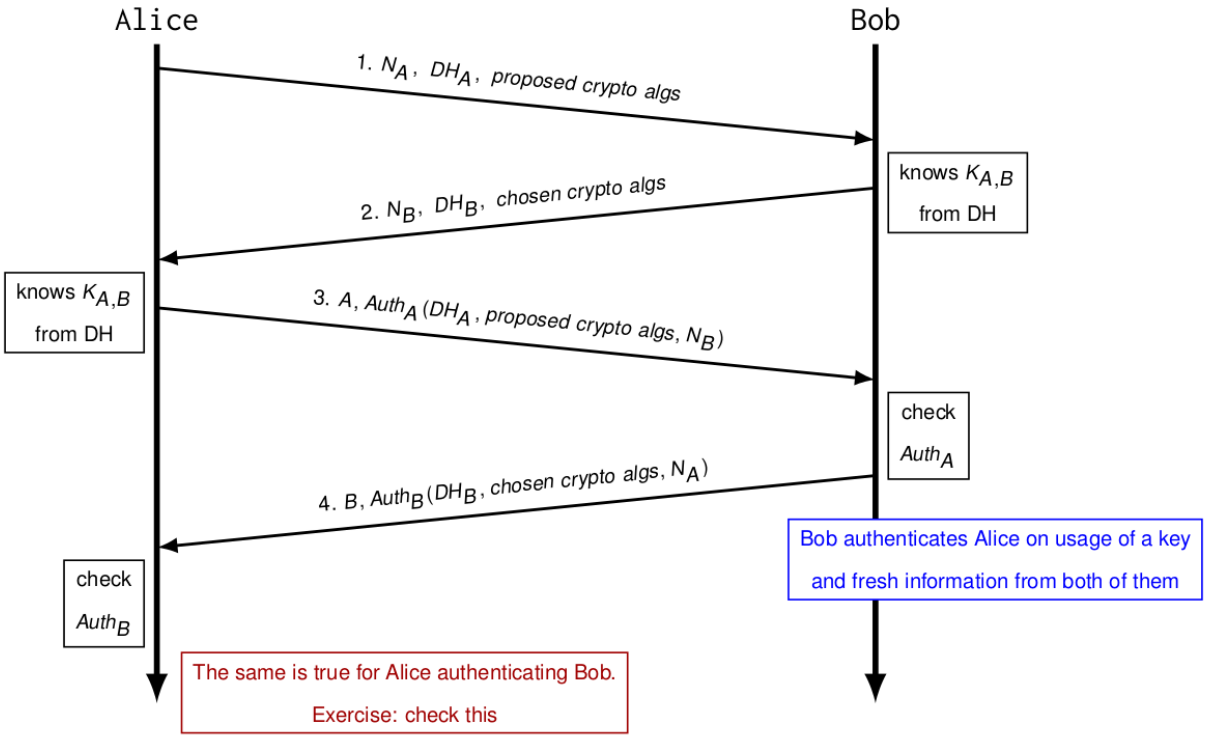


Figure 23: Protocol simplified

Messages 1 and 2 are only used for information exchange. 3 and 4 are used for authentication and if any verification fails, communication has to be aborted.

8.3 Needham Schroeder Protocol (NSP)

NSP is a protocol for mutual authentication and key establishment over an insecure network. There is a symmetric key version and a public-key version of it.

8.3.1 Symmetric Key NSP

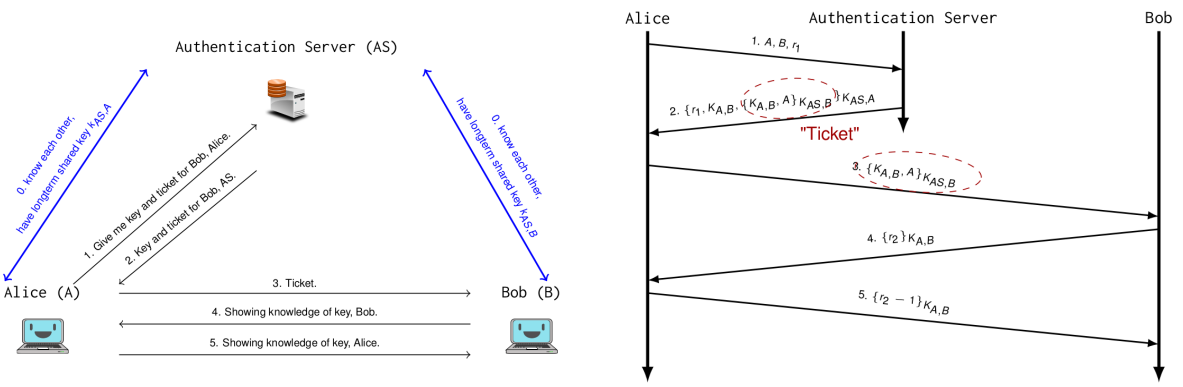


Figure 24: Symmetric NSP

r_i are random values here. Note here that NSP does not provide forward secrecy.

8.3.2 Public Key NSP

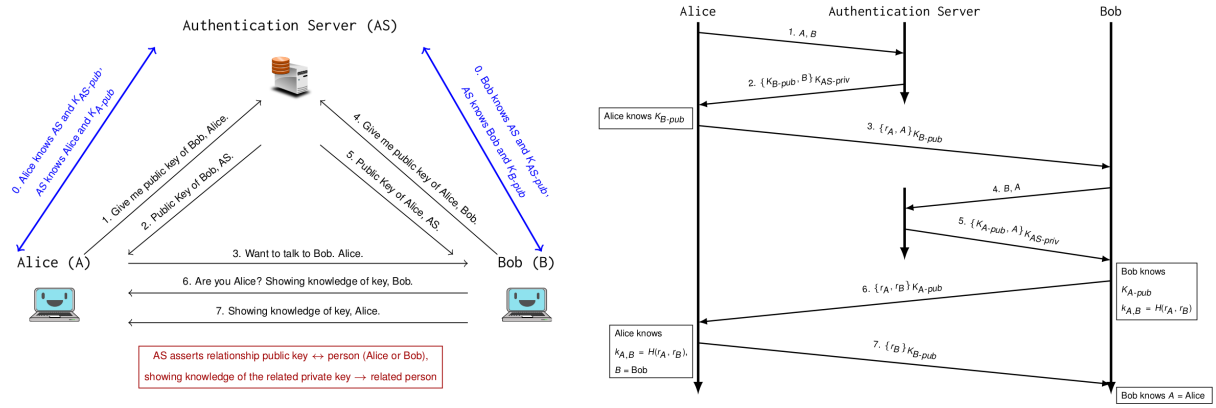


Figure 25: Asymmetric NSP

We use $\{\cdot\}$ for encryption and signing here, never mix them up in practice!

Attack on the Public key NSP

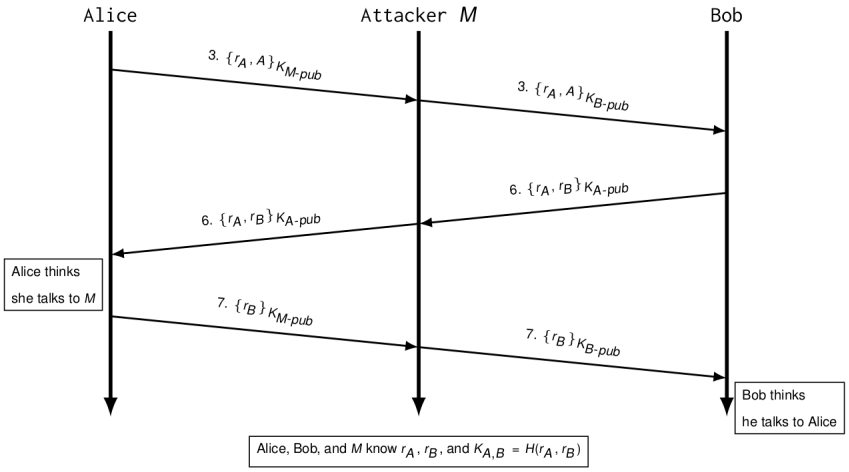


Figure 26: Asymmetric NSP Attack

The idea of the attack is that the attacker M tricks Alice to communicate with him and makes Bob believe, that he talks to Alice.