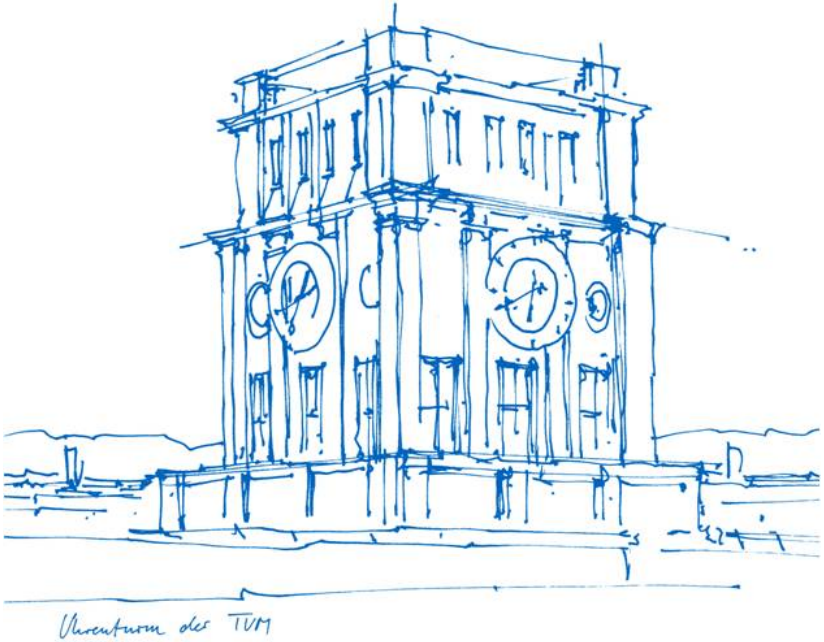
10 Cryptography 2

Heiko Niedermayer,

Georg Carle

Acknowledgements:
Günter Schäfer,
Benjamin Hof



Acknowledgements

- Günter Schäfer (TU Ilmenau), some slides still from the original network security lecture developed at KIT (then TU Karlsruhe)
- Benjamin Hof: inspirations from ilab Security slides

Overview

- Formalizing Integrity Protection
- Recap MAC
- Public Key Cryptography
  - RSA Cipher
  - Scheme for Confidentiality
  - Scheme for Integrity
- Hybrid Encryption Scheme
- Diffie-Hellman

Integrity Protection

Integrity protection has two functions

- Sign
  - $m, \sigma := sign_k(m)$
  - Generates protection
  - Signature / MAC: σ
- Verify
  - $verify_k(m, \sigma)$
  - Returns Boolean (1 = True, 0 = False)
  - Specifies check of integrity

Formalizing of Integrity Protection – Chosen Message Attack

Challenger $\mathcal{C}$ | Adversary $\mathcal{A}$

$k \leftarrow Gen(1^n)$ | $input\ 1^n$

$\xleftarrow{\quad m \quad}$

$\xrightarrow{\quad m\ \ ,\sigma \quad}$

Adversary can ask for a polynomial amount of examples where it selects the message.

...

Then it tries to forge a new message m_0 that was not yet signed by the challenger

output (m_0, σ_0)

Adversary $\mathcal{A}$ succeeds if and only if $verify_k(m_{\mathbf{0}}, \sigma_0)$=1

The goal of this game is to successfully forge a message.

Formalizing Integrity Protection - Discussion

- Adversary has to come up with a matching signature
- Guessing
  - n bits of hash /signature / MAC length
    → guessing has 2^{-n} chance to hit
- Adversary wins if it wins with chance significantly larger than 2^{-n}
- Protection scheme secure under the model if adversary wins only with chance $2^{-n} + \varepsilon$ and $small\ \varepsilon > 0$

Recap: MAC

Lets assume

$$sign_k(m) = m, (sha3(m) \oplus sha3(k))$$

- How does verify look like?

$$verify_k(m, \sigma) := \sigma = \left(sha3(m) \oplus sha3(k)\right)$$

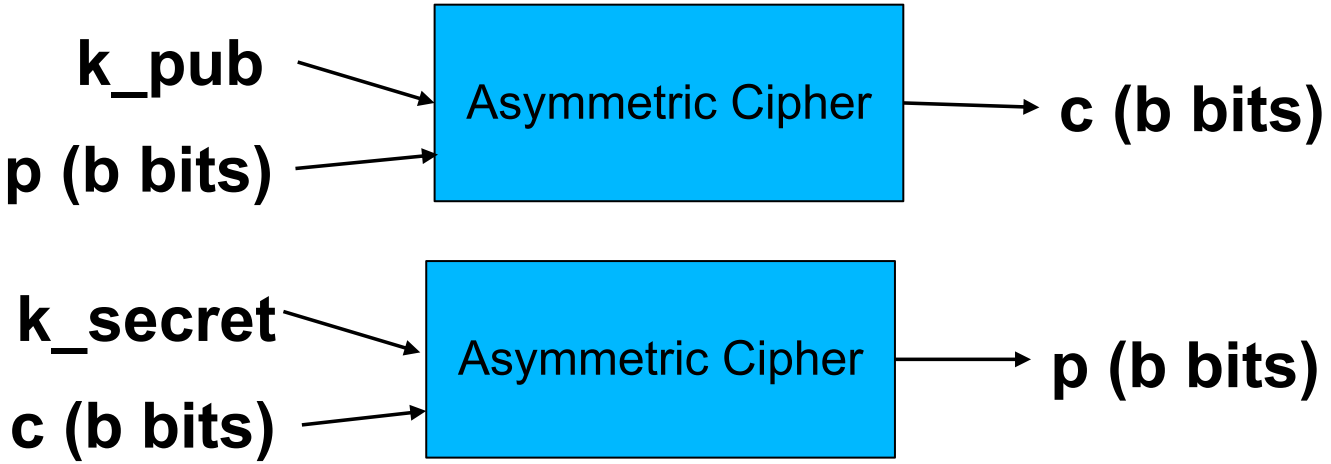
- Is it secure under the model?
  - No, attacker can send message m, compute sha3(m) and compute sha3(k) from the σ returned by the challenger
  - It can then forge the σ' for an m'.

What about:

$$sign_k(m) = m, HMAC - SHA256(m, k)$$

$$sign_k(m) = m, AES - XCBC - MAC(m, k)$$

Public Key Cryptography



- Outline
  - RSA
  - ECC
  - Hybrid Encryption
  - Diffie-Hellman

Some Mathematical Background for RSA

Definition: *<u>Euler's Φ Function:</u>*

Let $\Phi(n)$ denote the number of positive integers $m < n$, such that m is relatively prime to n.

→ "*m* is relatively prime to *n*" = the greatest common divisor (gcd) of *m* and *n* is one.

Let *p* be prime, then {1,2,…,p-1} are relatively prime to p, $\Rightarrow \Phi(p) = p-1$

Let *p* and *q* be distinct prime numbers and $n = p \times q$, *then*

$$\Phi(n) = (p-1) \times (q-1)$$

<u>Euler's Theorem:</u>

Let *n* and *a* be positive and relatively prime integers,

$$\Rightarrow a^{\Phi(n)} \equiv 1 \text{ MOD } n$$

The RSA Public Key Algorithm

RSA Key Generation:

Randomly choose *p, q* distinct and large primes
(really large: hundreds of bits = 100-200 digits each)

Compute $n = p \times q$, calculate $\Phi(n) = (p-1) \times (q-1)$ *(Euler's* Φ *Function)*

Pick $e \in Z$ such that $1 < e < \Phi(n)$ and *e* is relatively prime to $\Phi(n)$,
i.e. $\gcd(e,\Phi(n)) = 1$

Use the extended Euclidean algorithm to compute *d* such that

$$e \times d \equiv 1 \text{ MOD } \Phi(n)$$

The public key pk is (n, e)

The secret key sk is (n, d).

The RSA Public Key Algorithm

Definition: RSA function

Let p and q be large primes; let $n = p \times q$.
Let $e \in$ N be relatively prime to $\Phi(n)$.

Then RSA(e,n) := $x \rightarrow x^e$ MOD n

Example:

Let M be an integer that represents the message to be encrypted, with M positive, smaller than n.

To encrypt, compute: $C \equiv M^e$ MOD n

Decryption:

To decrypt, compute: $M' \equiv C^d$ MOD n

The RSA Public Key Algorithm

Why does RSA work:

As $d \times e \equiv 1$ MOD $\Phi(n)$

$\Rightarrow \exists$ k $\in$ Z: $(d \times e) = 1 + k \times \Phi(n)$

We sketch the "proof" for the case where M and n are relatively prime

$M' \equiv C^d$ MOD n

$\equiv (M^e)^d$ MOD n

$\equiv M^{(e \times d)}$ MOD n

$\equiv M^{(1 + k \times \Phi(n))}$ MOD n

$\equiv M \times (M^{\Phi(n)})^k$ MOD n

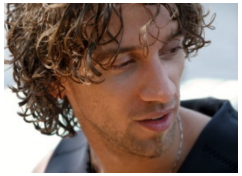
$\equiv M \times 1^k$ MOD n (Euler's theorem*)

$\equiv M$ MOD n = M

In case where M and n are not relatively prime, Euler's theorem can not be applied.

RSA for Confidentiality

Knows her public key, her secret key, and Bob‘s public key



Knows his public key, his secret key, and Alice‘s public key

- RSA assumption for confidentiality:
  - If we chose a random x and compute $\mathrm{c} = x^e mod\ n$
  - Then x cannot be recovered (~ relation between c and x looks random enough when keys unknown).
- Alice wants to send x to Bob.
  - She knows his public key (d_{Bob}, n_{Bob})
  - She computes c := $x^{d_{Bob}}\ mod\ n_{Bob}$
  - She sends c to Bob. He calculates $c^{e_{Bob}} mod\ n_{Bob}$ = x.

Chosen Plaintext Attack

Challenger $\mathcal{C}$

Adversary $\mathcal{A}$

$pk, sk \leftarrow GenRSAModulus(1^n)$

pk

m

$c \coloneqq Enc_{pk}(m)$

c

Note that adversary can calculate c herself in case of CPA and asymmetric encryption. Still the scheme should not leak information so that $\mathcal{A}$ can determine the correct b'. Deterministic schemes will fail.

...

m_0, m_1

m_0, m_1 of equal size selected by adversary

$b \leftarrow \{0,1\}$ $c \coloneqq Enc_{pk}(m)$

c

output guess b'

Chosen Ciphertext Attack

Challenger $\mathcal{C}$ | Adversary $\mathcal{A}$

$pk, sk \leftarrow GenRSAModulus(1^n)$

$\xrightarrow{pk}$

$\xleftarrow{c}$

$m := Dec_{sk}(m)$

$\xrightarrow{m}$

...

$\xleftarrow{m_0, m_1}$

m_0, m_1 of equal size selected by adversary

$b \leftarrow \{0,1\}$ $c := Enc_{pk}(m)$

$\xrightarrow{c}$

output guess b'

RSA for Confidentiality

- Pure use of Textbook RSA is deterministic
- Adversary can send m_0, m_1 as chosen plaintext and then resend them.
- Other issues
  - What happens with m=0 ? → c = 0 ?
  - What happens when $m^e < n$? → $c = m$?
- To achieve confidentiality, we have to use the correct encryption scheme containing the RSA algorithm as its basis.
- In the context of RSA, these schemes are called Padding Schemes.
  - E.g. PKCS, OAEP
  - They add random bits (non-determinism) and tend to avoid inputs like 0.

RSA-OAEP (Optimal asymmetric encryption padding)

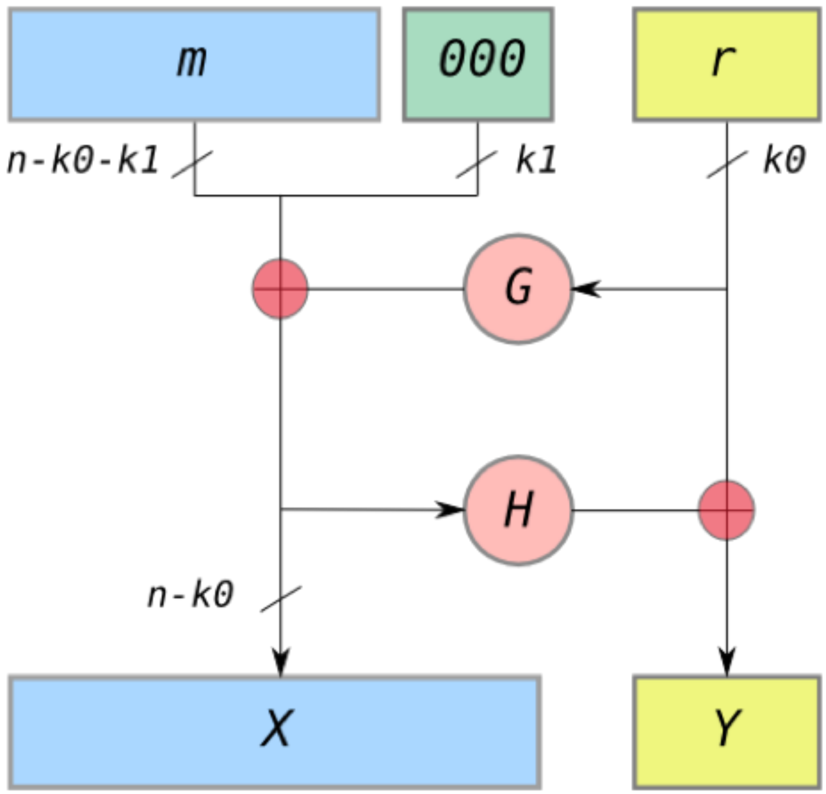
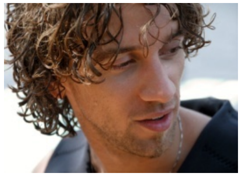


Figure taken from Wikipedia, Creative Commons https://de.wikipedia.org/wiki/Optimal_Asymmetric_Encryption_Padding

- G, H are hash functions
- Note that n in the figure refers to the bitlength of the RSA modulus.
- $\widehat{m}: == X||Y \;\; c := \widehat{m}^{d_{Bob}} mod\; n_{Bob}$

RSA for Integrity

Knows her public key, her secret key, and Bob‘s public key



Knows his public key, his secret key, and Alice‘s public key

- If the private key is used for encryption, anyone knowing the public key can decrypt.
- But what is the verify function?
  - If bits are flipped, it just decrypts to something else…
- Basic scheme
  - Alice uses a cryptographic hash function h and computes h(m)
  - She then encrypts h(m) with her secret key $\sigma \leftarrow Enc_{sk_{Alice}}$(h(m))
  - She send m, σ
  - Bob verifies that h(m) = $Dec_{pk_{Alice}}(\sigma)$

RSA-PSS

- There are dedicates signature schemes for RSA, e.g. RSA-PSS
- RSA-PSS hashes the message twice, adds padding, adds salt, and fills up the necessary bits
- The result is then encrypted with the secret key.
- It is part of the PKCS standards.

Hybrid Encryption Schemes

Knows her public key, her secret key, and Bob‘s public key

Knows his public key, his secret key, and Alice‘s public key

- Public Key cryptography is very expensive, many orders of magnitude slower than symmetric encryption or hashing.
- Hybrid encryption scheme
  - Alice protects shared symmetric key k with Bob’s public key
  - Alice then encrypts the large message with symmetric key k.

Hybrid Encryption Schemes / Key Agreement – Diffie Hellman

- Instead of Alice sending the symmetric key, a protocol could be used to generate a shared key between Alice and Bob.
- This Key Agreement is part of a larger protocol that usually
  - Authenticates the entitites
  - Provides additional protections for the communication
  - Keys are generated from result of Key Agreement via a Key Derivation Function (KDF)
- The Diffie-Hellman protocol is a public key scheme for key agreement.

Diffie-Hellman, Some Mathematical Background

Theorem/Definition: *primitive root, generator*

Let p be prime. Then $\exists$ g $\in$ {1,2,…,p-1} such that

$\{g^a \mid 1 \leq a \leq (p\text{-}1)\} = \{1,2,\ldots,p\text{-}1\}$ if everything is computed MOD p

i.e. by exponentiating g you can obtain all numbers between 1 and (p -1)

g is called a primitive root (or generator) of {1,2,…,p-1}

Example: Let p = 7. Then 3 is a primitive root of {1,2,…,p-1}

$1 \equiv 3^6$ MOD 7, $2 \equiv 3^2$ MOD 7, $3 \equiv 3^1$ MOD 7, $4 \equiv 3^4$ MOD 7,

$5 \equiv 3^5$ MOD 7, $6 \equiv 3^3$ MOD 7

Diffie-Hellman, Some Mathematical Background

Definition: discrete logarithm

Let *p* be prime, *g* be a primitive root of {1,2,…,p-1} and *c* be any element of {1,2,…,p-1}. Then $\exists$ *z* such that: $g^z \equiv c$ MOD *p*

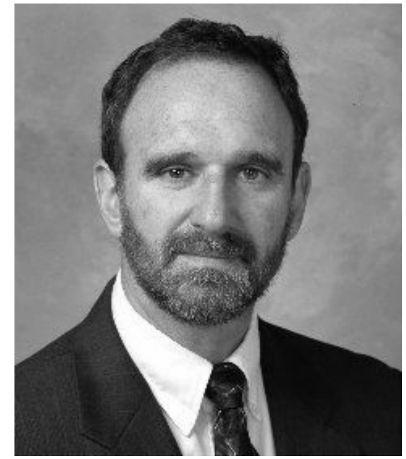
z is called the discrete logarithm of *c* modulo *p* to the base *g*

Example: 6 is the discrete logarithm of 1 modulo 7 to the base 3 as $3^6 \equiv 1$ MOD 7

The calculation of the discrete logarithm *z* when given *g*, *c*, and *p* is a computationally difficult problem and the asymptotical runtime of the best known algorithms for this problem is exponential in the bit-length of p

Diffie-Hellman Key Exchange (Textbook version)

Whitfield Diffie



Martin E. Hellman

Generate random
$a < p$

Compute
$X = g^a$ MOD p

Compute
$K = Y^a$ MOD p

(p, g, X) →

← Y

Generate random
$b < p$

Compute
$Y = g^b$ MOD p

Compute $K = X^b$
MOD p

Diffie-Hellman

- The DH construction contains insecure weak values,
  - g = 1, a = 0, b=0
  - Certain combinations of g and p
  - While Alice and Bob may try to avoid them, an attacker might not.
- ECC DH is Diffie-Hellman based on Elliptic Curves.

Perfect Forward Secrecy

- Assumption: for every new session a new DH key is generated and old keying material is deleted.
- Consequence: An attacker that has
  - Eavesdropped all messages
  - Broken a longterm key that protected the messages (e.g. Bob's private key)
    → Can now read the plaintext of the session establishment
  - Still, it cannot obtain the session key because the agreement is protection with DH (= an additional layer of cryptography that the attacker would need to break, hard due to DLog)
    →The attacker cannot decrypt the messages of the session.

RSA vs ECC vs Symmetric vs Hash Functions

- Elliptic Curve Cryptography (ECC) is a variant of Public Key cryptography that is based on elliptic curves
- ECC requires less bits to achieve to achieve a similar security as RSA
- ECC is usually more efficient than RSA
- Key length and security level
  - “Similar” level: 256 bits ECC vs 3072 bits RSA / DH vs 128 bits Symmetric Key Crypto vs 256 bits Cryptographic Hash Function (output length)
  - For Diffie-Hellman, normal Dlog similar to RSA, ECC Dlog similar to ECC
  - For key with the long-term use you should use significantly larger key size.