Introduction to

Cryptography

Lecture 19-21

©Michael Luttenberger

Chair for Foundations of Software Reliability and Theoretical Computer Science
Technische Universität München

2019/01/22, 14:26

1 Lectures 19-21 – Public Key Cryptography, DLP-based PKES, Signatures

Motivation

- Main drawbacks of private-key ES and MACs:
  - (i) Key distribution:

    Either all members of a party meet in person from time to time in order to generate a new secret key;

    or one of them generates the key and securely transfers it to the others.
  - (ii) Key storage (Number of keys):

    n parties need $\binom{n}{2}$ secret keys in total;

    every party needs to store $n-1$ keys securely.
  - (iii) Lack of identity authentication w.r.t. to a third party:

    Every member of a party can compute a MAC; MAC only identifies the party as a whole.

▷ Private-key schemes impractical for system with an a-priori unknown user base/transient interactions (e.g. on-line sales company).

Motivation

- Basic idea of public-key encryption:

  Try to implement a “Post-office box”.

  - Private key "=" key to the P.O. box.

    Private key gives the unique ability to open the P.O. box.

    (Unless the key is stolen from him and/or duplicated ...)

  - Public key "=" address of the P.O. box

    To send a letter to Bob, look up the address of Bob’s P.O. box in a public directory and put the letter directly into it.

    (As long as the integrity of the directory is guaranteed ...)

  - Key generator "=" setup a new P.O. box

    (Make sure you create indeed a new P.O. box ...)

- Basic idea of "public-key MAC":

  "Reverse" of a P.O. box ("vending machine").

Motivation

- One important difference though between public-key ES and P.O. boxes:

▷ You need to rent a P.O. box at your post office.

In particular, you get a unique key and P.O. box from the post office.

▷ In a PKES, you generate the private and public key yourself.

Ideally – except for negligible prob. – nobody except for you will ever have access to the private key.

▷ So, a PKES allows you to generate your own "'P.O. box" and private key, and pass the P.O. box then to the post office.

Motivation

- The private key and the ability bestowed by it identify Bob (except for negl. prob.) and its secrecy has only to be guaranteed by Bob.
- The public key is given to everyone, e.g., stored in a public directory.

▷ Problem: Who makes sure that all entries of the directory are correct?

▷ Later: Introduce a certificate authority (CA) which

  - is trusted by everyone (!),
  - guarantees the secure distribution of its own public keys,
  - certifies the correctness of the public keys of the other users.

From private-key to public-key crypto

- Little conceptual changes required:

▷ Gen now creates both a private and a public key.

▷ Security definitions stay almost the same, except:

Eve is given the public key.

▷ Some name changes:

ES $\rightsquigarrow$ PKES,

MAC $\rightsquigarrow$ digital signatures.

▷ Decryption is allowed to fail with negligible probability.

This reflects e.g. that we might mistake a composite for a prime when using the Miller-Rabin test.

Public-key encryption schemes

- **Definition:** A public-key encryption scheme (PKES) consists of PPT-algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ s.t.:

Algorithm	Type	Input	Output
Gen	PPT	1^n	$I \overset{r}{\in} \mathcal{I}_n$ with $\lvert I\rvert \geq n$
Enc	PPT	$\mathsf{ek}_I \in \mathcal{K}_n^{\mathsf{Enc}}, m \in \mathcal{M}_{\mathsf{ek}}$	$c \overset{r}{:=} \mathsf{Enc}_{\mathsf{ek}_I}(m) \in \mathcal{C}_{\mathsf{ek}}$
Dec	DPT	$\mathsf{dk}_I \in \mathcal{K}_n^{\mathsf{Dec}}, c \in \mathcal{C}_{\mathsf{ek}}$	$m = \mathsf{Dec}_{\mathsf{dk}_I}(c) \in \mathcal{M}_{\mathsf{ek}}$

Gen, on input 1^n, outputs key parameters I.

The parameteres I determine the public encryption key ek_I and the private decryption key dk_I.

  - We assume that $\lvert\mathsf{ek}_I\rvert, \lvert\mathsf{dk}_I\rvert \geq n$.
  - ek_I determines the set of plain- and ciphertexts: $\mathcal{M}_{\mathsf{ek}_I}, \mathcal{C}_{\mathsf{dk}_I}$.
  - $\Pr_{I \overset{r}{:=} \mathsf{Gen}(1^n), m \overset{u}{\in} \mathcal{M}_{\mathsf{ek}_I}} [\mathsf{Dec}_{\mathsf{dk}_I}(\mathsf{Enc}_{\mathsf{ek}_I}(m)) \neq m]$ has to be negligible.
  - "deterministic" and "stateless/stateful" defined as for general ES.

Public-key encryption schemes

- We have already discussed how OWFs based on integer factorization and the DLP can be obtained.

▷ This usually envolves generating specific objects like prime numbers.

▷ For instance, when using the Miller-Rabin test, we might mistake a composite for a prime with some negligible, but non-zero probability.

▷ In these cases, the algebraic assumptions underlying a public-encryption scheme need not be satisfied s.t. decryption might not yield the original plaintext anymore.

▷ Hence, we allow that for PKES decryption fails with negligible probability.

- We only consider stateless PKES in the following, and therefore only pass ek to Eve in the following definitions.

CPA-security for PKES

- **Definition**: Game $\textsc{IndCPA-PK}$

1. Bob runs $\mathsf{Gen}(1^n)$ to obtain I and passes $\mathsf{ek} := \mathsf{ek}_I$ to Eve.
2. Eve runs $\mathcal{A}(\mathsf{ek})$ to obtain messages sequences $(m_0^{(1)}, \dots, m_0^{(t)})$ and $(m_1^{(1)}, \dots, m_1^{(t)})$ with all messages in $\mathcal{M}_{\mathsf{ek}}$ and $\left|m_0^{(i)}\right| = \left|m_1^{(i)}\right|$.
3. Bob chooses $b \overset{u}{\in} \{0,1\}$ and computes $c^{(i)} \overset{r}{:=} \mathsf{Enc}_{\mathsf{ek}}(m_b^{(i)})$.
4. Eve runs $\mathcal{A}(\mathsf{ek}, c^{(1)}, \dots, c^{(t)})$ to obtain r.

▷ Let $\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCPA-PK}}(\mathcal{A})$ be the event that $r = b$.

A PKES $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable (multiple) encryptions under a chosen-plaintext attack (IND-CPA) (short: is CPA-secure) if for every PPT-attack $\mathcal{A}$ its advantage in winning the game $\textsc{IndCPA-PK}$ is negligible, i.e.:

$$\left| \Pr\left[\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCPA-PK}}(\mathcal{A})\right] - \frac{1}{2} \right| \text{ is negligigble in } n.$$

CCA-security for PKES

- **Definition**: Game $\textsc{IndCCA-PK}$

1. Bob runs $\mathsf{Gen}(1^n)$ to obtain I and passes to Eve both $\mathsf{ek} := \mathsf{ek}_I$, and $\mathsf{Dec}_{\mathsf{dk}_I}$ in a black box (decryption oracle).
2. Eve runs $\mathcal{A}^{\mathsf{Dec}_{\mathsf{dk}_I}}(\mathsf{ek})$ to obtain messages sequences $(m_0^{(1)}, \ldots, m_0^{(t)})$ and $(m_1^{(1)}, \ldots, m_1^{(t)})$ with all messages in $\mathcal{M}_{\mathsf{ek}}$ and $\left|m_0^{(i)}\right| = \left|m_1^{(i)}\right|$.
3. Bob chooses $b \overset{u}{\in} \{0,1\}$ and computes $c^{(i)} \overset{r}{:=} \mathsf{Enc}_{\mathsf{ek}}(m_b^{(i)})$.
4. Eve runs $\mathcal{A}^{\mathsf{Dec}_{\mathsf{dk}_I}}(\mathsf{ek}, c^{(1)}, \ldots, c^{(t)})$ to obtain r where $\mathcal{A}$ must not query $\mathsf{Dec}_{\mathsf{dk}_I}$ for any $c^{(i)}$.

▷ Let $\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCCA-PK}}(\mathcal{A})$ be the event that $r = b$.

A PKES $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable (multiple) encryptions under an adaptive chosen-ciphertext attack (IND-CCA2) (short: is CCA-secure) if for every PPT-attack $\mathcal{A}$ its advantage in winning the game $\textsc{IndCCA-PK}$ is negligible, i.e.:

$$\left|\Pr\left[\mathsf{Win}_{n,\mathcal{E}}^{\textsc{IndCCA-PK}}(\mathcal{A})\right] - \frac{1}{2}\right| \text{ is negligigble in } n.$$

Security for PKES

- Weak CCA-security (IND-CCA1): Eve must not use the decryption oracle once c is obtained.

  Also called lunchtime attack: decryption oracle may only be used while Bob is at lunch.

- The definition of CPA- and CCA-security in the public-key setting are essentially the same as in the private-key setting:

  The only difference is that Eve is given directly the public ek.

  Hence, no encryption orcale is required.

Security for PKES

- ▷ Many results carry over:
- • **Ex**: No stateless and deterministic PKES can be CPA-secure.
- • Equivalent semantic notions of security exist. See [24].
- ▷ **Theorem**: A PKES is CPA-secure resp. CCA-secure iff it is already CPA-secure resp. CCA-secure when Eve is restricted to a single message pair ($t = 1$). (See e.g. [29] p. 345 for a proof.)
- ▷ **Corollary**: We can build from a CPA-secure PKES with finite message space a CPA-secure PKES which can encrypt messages of unbounded length.
- ▷ But, as in the private-key setting, this construction does not preserve CCA-security.

Security for PKES

- Notions of CPA- and CCA-security do not capture attacks where Eve actively attacks the communication between Alice and Bob.

  For instance, Eve could try to intercept Bob's public key $\mathsf{ek}^{\mathsf{Bob}}$ on its way to Alice, and replace it by $\mathsf{ek}^{\mathsf{Eve}}$.

▷ We must ensure the integrity of the "phone book" relating public keys to entities:

  Public key infrastructure (PKI) based on digital signatures and certificates.

- For PKES, CCA-security is regarded as the more appropriate definition of security:
  - E.g., a webserver which automatically replies to all received messages and therefore may act as a decryption oracle to an attacker. (See here.)

CCA-security for PKES

- [20] shows that CCA-security is equivalent to non-malleability of ciphertexts under CCA:
  - Assume Alice sends orders for buying/selling stocks to Bob.
  - Semantic security guarantees that Eve cannot obtain any additional information from c;
  - but assume that Bob also does business with Eve.
  - Then Eve can also interact with Bob, thus, knows the message format used for orders, and

    therefore can try to change c in such a way that “buy” becomes “sell” or that the stated quantity changes.

▷ Non-malleability guarantees that PPT-Eve cannot obtain from an intercepted ciphertext c a new ciphertext which is related to c in a way determined by Eve – except with negligible prob.

The public-key "revolution"

- In 1976, Whitfield Diffie and Martin Hellman published a paper "New Directions in Cryptography" [19]:

▷ The two are credited to be be the first to point out the impact of public-key schemes on public communication,

  - Before, cryptography was regarded to be only of use for military/intelligence agency or highly secret business communication because of the need to securely distribute/share the secret key.

  and to publish the first practical public-key scheme for public key exchange (DH protocol).

  - Diffie and Hellman themselves give also credit to Merkle, but the protocol had already been discovered and kept secret by Williamson in 1974 working at the British intelligence agency "Government Communications Headquarters". (See here.)
  - Nowadays, the DH protocol is part of several larger protocols, and is used as key encapsulating mechanism in hybrid PKES.

Diffie-Hellman protocol

- Goal: Share a secret k over a public channel.
- **Definition:** (basic version)

  Let $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ be a DLP-generator.

  1. Bob runs $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}(1^n)$ to obtain $I = (\mathbb{G}, q, g)$, i.e., a (description of a) cyclic group $\mathbb{G}$ of order q generated by g.
  2. Bob generates $b \overset{u}{\in} \mathbb{Z}_q$ and computes $h_B := g^b$.

     He sends $(\mathbb{G}, q, g, h_B)$ to Alice.
  3. Alice generates $a \overset{u}{\in} \mathbb{Z}_q$ and computes $h_A := g^a$.

     She sends h_A to Bob.
  4. Alice and Bob obtain the shared secret $k := g^{ab} = h_A^b = h_B^a$.
- In practice, $a, b \neq 0$.
- See [19] for the multi-party extension.

Diffie-Hellman protocol: Requirements

- Necessary requirements for the secrecy of k:

▷ Authenticate the origin of a message/identity of the sender.

  ▷ **Ex**: Describe a man-in-the-middle attack under the assumption that Alice and Bob do not verify the origin of a message.

  - Problem: Using a MAC would require Alice and Bob to already have shared a secret key.

▷ The DLP has to be hard relative to the used $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$.

  - Otherwise compute b from $(\mathbb{G}, q, g, h_B)$, and subsequently $k = h_A^b$.

▷ But this is not enough:

  - PPT-Eve must not be able to compute g^{ab} from $(\mathbb{G}, q, g, h_A, h_B)$

  ▷ Called computational Diffie-Hellman (CDH) problem.

  ▷ Might be easier then computing a or b (later).

Diffie-Hellman protocol: Remarks

- What to do with the shared secret group-element $k = g^{ab}$?

▷ Original proposal by Diffie and Hellman:

Use it as a key for a private-key ES(+MAC) but:

- Private-key schemes usually require some $k' \overset{u}{\in} \{0,1\}^n$.
- g^{ab} is some (more or less) random group element.
- Eve knows g^a and g^b, and, thus, has some information on k.

▷ Might hope to use $k = g^{ab}$ as seed for the Blum-Micali-PRG.

▷ In practice: hash k to some binary string k' (later).

▷ Elgamal's solution:

Use $k = g^{ab}$ directly as a (pseudorandom) one-time pad.

- Alice sends h_A and $c := h_B^a \cdot m = k \cdot m$ for $m \in \mathbb{G}$.
- Bob computes $h_A^{|\mathbb{G}|-b} \cdot c$.

Elgamal PKES

- **Definition**: Let $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ be a DLP-generator.

  Gen: On input 1^n, generates $I = (\mathbb{G}, q, g, x, h)$ where

  - $(\mathbb{G}, q, g)$ is obtained by running $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}(1^n)$.
  - $h = g^x$ with x chosen uniformly at random from $\mathbb{Z}_q$.
  - En-/deccryption key: $\mathsf{ek}_I = (\mathbb{G}, q, g, h)$, $\mathsf{dk}_I = (\mathbb{G}, q, g, x)$
  - Admissible plain-/ciphertexts: $\mathcal{M}_{\mathsf{ek}_I} = \mathbb{G}$, $\mathcal{C}_{\mathsf{ek}_I} = \mathbb{G} \times \mathbb{G}$

  Enc: Given ek and $m \in \mathcal{M}_{\mathsf{ek}}$, choose $y \overset{u}{\in} \mathbb{Z}_q$, and output $c = (g^y, h^y \cdot m)$.

  Dec: Given dk and $c = (u, v) \in \mathcal{C}_{\mathsf{ek}}$, output $v \cdot u^{q-x}$.

- $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ is allowed to generate with negl. prob. key parameters not satisfying these assumption.
- In practice, $x, y \neq 0$ (prob. that $x = 0 \vee y = 0$ is negl.).

Security of the DH protocol and Elgamal

- As in the DH protocol, we have to assume that Eve does not attack the communication between Alice and Bob itself.
- **Lemma**: Elgamal is not CCA-secure.

▷ Proof: If $c = (u, v)$ is a ciphertext,

then $(u, v \cdot x)$ is again an admissible ciphertext for any $x \in \mathbb{G}$

which Eve may now decrypt using her oracle.

▷ In other words: Elgamal is malleable.

- Same attack as for all variants of OTP (prOTP,CTR,OFB):

  there the group is $\langle \mathbb{Z}_2^n, \oplus, 0^n \rangle$, here it is some cyclic group $\langle \mathbb{G}, \cdot, 1 \rangle$.

▷ But this has also its uses, see e.g. homomorphic encryption.

Security of the DH protocol and Elgamal

- So, the best we can hope for is CPA-security of Elgamal.
  - But CCA-secure PKES related to Elgamal exist:

    Cramer-Shoup-PKES [18]: CCA-secure; DDH needs to be hard.

    Damgård-Elgamal-PKES: weak CCA-security; but DDH does not suffice.
- Certainly, the DLP needs to be hard relative to $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ in order to prevent Eve from computing a, b from g^a, g^b.

  But this does not suffice for CPA-security.

Security of the DH protocol and Elgamal

- As motivated, Elgamal can be understood as the DH protocol plus the idea to use the shared secret group element k as pseudorandom one-time pad.

▷ The decisional Diffie-Hellman problem formalizes exactly the requirement that $k = g^{ab}$ is pseudorandom, i.e.:

Although PPT-Eve knows the public information $(\mathbb{G}, q, g)$, g^b, and g^a, she cannot distinguish the secret g^{ab} from truely random $x \overset{u}{\in} \mathbb{G}$.

▷ That is: No PPT-distinguisher $\mathcal{D}$, given the public information $(\mathbb{G}, q, g, h_A, h_B)$, should be able to distinguish k generated by the DH protocol from a truly random group element $r \overset{u}{\in} \mathbb{G}$.

Decisional Diffie-Hellman problem

- **Definition**: Let $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ be a DLP-generator.

  The decisional Diffie-Hellman (DDH) problem is hard w.r.t. $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ if every PPT-distinguisher $\mathcal{D}$ wins only negl. better than prob. $1/2$ in:

  1. $(\mathbb{G}, q, g) \overset{r}{:=} \mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}(1^n)$.
  2. $g_1 := g^x$, $g_2 := g^y$ for $x, y \overset{u}{\in} \mathbb{Z}_q$.
  3. $b \overset{u}{\in} \{0, 1\}$.
  4. If $b = 0$: $z \overset{u}{\in} \mathbb{Z}_q$,$g_3 := g^z$; (“perfect world”: $g_3 \overset{u}{\in} \mathbb{G}$)

     If $b = 1$: $g_3 := g^{xy}$. (“real world”: $g_3 = g_1^y = g_2^x$ the shared secret)
  5. $c \overset{r}{:=} \mathcal{D}(\mathbb{G}, g, q, g_1, g_2, g_3)$.

  ▷ $\mathcal{D}$ wins iff $r = b$.

▷ See [12] for an overview on the results regarding the DDH problem.

CPA-security of Elgamal: Proof outline

- **Theorem**: If the DDH problem is hard relative to $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$,

  then the Elgamal PKES based on $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ is CPA-secure.

- Proof sketch:

  The proof works essentially the same as the one for the pseudorandom OTP.

  The distinguisher $\mathcal{D}$ gets $(\mathbb{G}, q, g, g_1, g_2, g_3)$,

  and simulates the CPA-game v.s. an attack on Elgamal.

  To this end it passes $(\mathbb{G}, q, g, g_1)$ to $\mathcal{A}$ as public key,

  and encrypts m_b as $(g_2, g_3 \cdot m_b)$.

▷ **Ex**: Complete the proof.

The DDH problem vs. the DLP

- Obviously:

  DDH hard w.r.t. $(\mathbb{G}, q, g) \Rightarrow$ DLP hard w.r.t. $(\mathbb{G}, q, g)$.

▷ But there are groups for which the DDH is easy, while the DLP is still conjectured to be hard [28]:

▷ For instance, consider $\mathbb{G} = \langle g \rangle = \mathbb{Z}_p^*$ with p prime.

  - Let $a, b, r \overset{u}{\in} \mathbb{Z}_{p-1} = \{0, 1, \dots, p-2\}$.

  ▷ Then a is even, i.e., g^a a square with prob. $1/2$.

  ▷ So, ab is even, i.e., g^{ab} a square with prob. $3/4$.

  - Given (g^a, g^b, g_3) we can distinguish $g_3 = g^{ab}$ from $g_3 = g^r$ by testing whether g_3 is a square or not.

  ▷ Simply compute the Legendre symbol $\left(\frac{g_3}{p}\right) = {g_3}^{\frac{p-1}{2}} \bmod p$.

The DDH problem vs. the DLP

- We can prevent this way of distinguishing g^{ab} from g^r by moving to the subgroup of quadratic residues modulo safe primes: $\mathbb{QR}_p \le \mathbb{Z}_p^*$.

▷ Or more general, use strong primes:

▷ **Conjecture**: Let p, q be primes such that $p = kq + 1$ for some $k \in \mathbb{N}$, and $q > p^{1/10}$, i.e., "q almost as large as p".

For $\langle g \rangle = \mathbb{Z}_p^*$, the DDH problem is hard w.r.t. $\langle g^k \rangle$.

- Note that k is always even, so $\langle g^k \rangle \le \mathbb{QR}_p$, and $\left|\langle g^k \rangle\right| = q$.
- See [12, 28] for more examples, in particular, w.r.t. elliptic curves.

The DDH problem vs. the DLP

- **Ex**: Give a CPA-attack on Elgamal when used with $\mathbb{Z}_p^*$.
- **Ex**: For a safe prime $p = 2q + 1$, we can easily compute a square root of a quadratic residue x by means of $x^{\frac{p+1}{4}} \bmod p$.

  Let $s(x)$ be the smaller of the two square roots of x modulo p.

  Determine the distribution of $s(g^{ab})$ for $a, b \overset{u}{\in} \mathbb{Z}_q$.
- **Ex**: Explicitly formulate Elgamal w.r.t. $\mathbb{QR}_p$ modulo a safe prime p.

  Use $s(x)$ from above for Dec.
- **Ex**: Let $\mathbb{G} = \langle g \rangle$ be any group of prime order q.

  Determine the distribution of g^{ab} for $a, b \overset{u}{\in} \mathbb{Z}_q$ in general.

The Computational Diffie-Hellman (CDH) problem

- Recall for the DH protocoll we need to require that

  Computing the secret g^{ab} from the public $(\mathbb{G}, q, g, g^a, g^b)$ is hard.

▷ **Defintion**: The CDH is hard relative to $\mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}$ if

every PPT-adversary $\mathcal{A}$ succeeds only with negl. prob. in:

1. $(\mathbb{G}, q, g) \overset{r}{:=} \mathsf{Gen}\mathbb{G}_{\mathsf{cyc}}(1^n)$.
2. $g_1 := g^x$, $g_2 := g^y$ for $x, y \overset{u}{\in} \mathbb{Z}_q$.
3. $r \overset{r}{:=} \mathcal{D}(\mathbb{G}, q, g, g_1, g_2)$.

   ▷ $\mathcal{D}$ wins iff $r = g^{xy}$ (in $\mathbb{G}$).

- “DDH hard $\Rightarrow$ CDH hard $\Rightarrow$ DLP hard” w.r.t. the same $(\mathbb{G}, q, g)$.
- “DLP hard $\Rightarrow$ CDH hard” is unknown.

▷ There are families of groups w.r.t. which “DLP $\equiv_{\mathsf{PPT}}$ CDH” but “DDH easy”. See [28].

Elgamal in practice

- Recall: $\mathcal{M}_{(\mathbb{G},q,g,h)} = \mathbb{G}$ is the message space of Elgamal.
- As Elgamal is CPA-secure, we can obtain from it a CPA-secure PKES for Σ^+.
  - Alice splits the message m into blocks $m^{(i)}$ which can be encoded in $\mathbb{G}$
  - ▷ She then chooses for every block $m^{(i)}$ a random element $a^{(i)} \overset{u}{\in} \mathbb{Z}_q$ and sends $g^{a^{(i)}}, h_B^{a^{(i)}} \cdot m^{(i)}$ to Bob.
- Downside:

  For sufficiently secure groups, a single exponentiation takes as much time as several thousand calls to a block cipher like AES.

▷ In practical PKES, Elgamal resp. the DH protocol is only used for exchanging a secret key between Alice and Bob which is then used for a private-key ES.

▷ This is called hybrid encryption (next lecture).

Hybrid encryption

- Hybrid encryption: key encapsulating mechanism (KEM) + data encapsulating mechanism (DEM) [34].
  - The DEM is simply a sufficiently secure private-key ES built,

    e.g. a AES-256-rCTR + AES-128-CBC-MAC (EtM).
  - The KEM is some way to exchange a secret key for the DEM over a public channel.

    e.g. generate $k \overset{u}{\in} \{0,1\}^n$, encode it as a group element, and transfer it using Elgamal, or

    e.g. use DH to share a secret group element, then derive a key from the shared secret group element e.g. by applying a hash function.
  - ▷ Key derivation function (KDF)
- El Gamal as hybrid encryption:

  KEM = DH, DEM = prOTP.
- Here: security definitions only for PKES as KEM.
  - ▷ For more general definitions see, e.g. [34].

Hybrid encryption: Using a PKES as KEM

- **Definition**: Given:
  - $\mathcal{E}^a = (\mathsf{Gen}^a, \mathsf{Enc}^a, \mathsf{Dec}^a)$ PKES (asymmetric)
  - $\mathcal{E}^s = (\mathsf{Gen}^s, \mathsf{Enc}^s, \mathsf{Dec}^s)$ ES (symmetric)
  - (Some efficient encoding of the keys of $\mathcal{E}^s$ as messages of $\mathcal{E}^a$.)

  Let $\mathcal{E}^h = (\mathsf{Gen}^h, \mathsf{Enc}^h, \mathsf{Dec}^h)$ be the PKES:

  Gen^h: simply Gen^a.

  Enc^h: on input ek and a plaintext m admissible w.r.t. $\mathcal{E}^s$:

  1. Choose $k \overset{u}{\in} \{0,1\}^n$.
  2. Output $c = (\mathsf{Enc}^a_{\mathsf{ek}}(k), \mathsf{Enc}^s_k(m))$.

  Dec^h: on input dk and ciphertext $c = (c_k, c_m)$:

  1. Compute $k := \mathsf{Dec}^a_{\mathsf{dk}}(c_k)$.
  2. Output $\mathsf{Dec}^s_k(c_m)$.

Hybrid encryption: Using a PKES as KEM

- Note that by definition of $\mathcal{E}^h$, for every new message a new key k is chosen by Enc^h. Thus:
- **Theorem**: (see [29], p. 351)

  If $\mathcal{E}^a$ is CPA-secure and $\mathcal{E}^s$ is comp. secret,

  then $\mathcal{E}^h$ is also CPA-secure.
- **Theorem**: (see [18, 26])

  If both $\mathcal{E}^a$ and $\mathcal{E}^s$ are CCA-secure, then $\mathcal{E}^h$ is also CCA-secure.
- Similar results for general KEMs, like the DH protocol [1, 34].

▷ Examples: DHIES (DEM $=$ DH protocol $+$ hashing, KEM $=$ CCA-secure ES) or ECIES (DHIES with EC as groups).

Example: ECIES-KEM

- ECIES-KEM is a variant of the DH-KEM using elliptic curves [34]:

  1. Bob chooses a (description of a) sufficiently secure subgroup of an elliptic curve, e.g., $\mathbb{G} = (p, a, b, P, N, h)$.

     He then chooses $s_B \overset{u}{\in} \mathbb{Z}_N \setminus \{0\}$ and computes $H_B := s_B \cdot P$, e.g., by squaring.

     He publishes his public key $(\mathbb{G}, H_B)$.

  2. For every new message m, Alice chooses $s_A \overset{u}{\in} \mathbb{Z}_N \setminus \{0\}$, and computes $H_A := s_A \cdot P$ and $K = (x_K, y_K) := s_A \cdot H_B$.

     She derives a secret DEM-key k by computing $\mathsf{KDF}(H_A || x_K)$.

     She sends H_A and an encryption of m to Bob.

  3. Bob computes $K = (x_K, y_K) := s_B \cdot H_A$, and obtains the DEM-key k by computing $\mathsf{KDF}(H_A || x_K)$.

DH protocol as KEM

- Recall: DH protocol allows to share a secret group element g^{ab} with $a, b \overset{u}{\in} \mathbb{Z}_{|\mathbb{G}|}$.

▷ So, g^{ab} is random (but not uniformly distributed) in $\mathbb{G}$.

▷ Goal: use some "randomness-extractor" K to derive from g^{ab} a truly random bit string $K(g^{ab})$ of sufficient length.

▷ In practice, K is (constructed from) a hash function, and called a key derivation function (KDF).

Key derivation functions (KDFs)

- Let $h\colon \{0,1\}^* \to \{0,1\}^n$ be a hash function.
- Depending on the required key lengths and number of keys output length n might be too small.
- Some constructions of KDFs from h if the output is too short:

  Fix some fixed-length encoding function $\lfloor\rceil\colon \mathbb{Z}_{2^l} \to \{0,1\}^l$.

  - Construction 1: $K(x) := h(\lfloor 0\rceil || x) || h(\lfloor 1\rceil || x) || \ldots || h(\lfloor L\rceil || x)$
  - Construction 2: $K(x) := h(x || \lfloor 0\rceil) || h(x || \lfloor 1\rceil) || \ldots || h(x || \lfloor L\rceil)$

  This stretches the output length from n to $L \cdot n$.

  - Alternatively use some hash function like Keccak which allows to choose the output length as needed.

DH protocol as KEM

- DH as KEM: apply KDF K to either $x = g^{ab}$ or $x = g^{a}||g^{ab}$ to derive the secret keys.
  - Encoding of g^{a}, g^{ab} is fixed by the protocol, see e.g. [1, 34].

▷ Additional advantage of the KDF: K obfuscates properties of x.

  - E.g.: Given p, g, g^{x}, g^{y} for p prime, $\langle g\rangle = \mathbb{Z}_p^*$, $x, y \overset{u}{\in} \mathbb{Z}_{p-1}$,

    it is unknown how to distinguish $K(g^{xy})$ from $z \overset{u}{\in} \{0,1\}^{L\cdot n}$.

  ▷ Legendre symbol does not seem to help anymore.

  ▷ $K(g^{xy})$ might still pass as pseudorandom, although g^{xy} does not.

  ▷ Might suffice that the CDH is still conjectured to be hard w.r.t. $\mathbb{Z}_p^*$.

▷ Problem: unknown how to prove this using only standard assumptions on hash functions like collision-resistance or (2nd) preimage resistance.

The random oracle model

- "Sanity check": random oracle model [6]

  Idealized setting where for the proof of security a hash function is treated as a RFO.

▷ Recall: A RFO chooses for every possible input x a truly random output $y \overset{u}{\in} \{0,1\}^n$.

  - **Ex**: If h is a RFO of output length n, then also the two KDF constructions yield RFOs of output length $L \cdot n$ (domain separation).

- For a RFO there is no relation between input and output.

▷ No properties of the input can carry over to the output.

▷ RFO captures the intuitive advantages of a hash function resp. KDF.

- **Problem**: In reality, the hash function/KDF is a deterministic algorithm which is known to everyone.

The random oracle model

- At least in the ROM, one can show that

  DH+KDF as KEM combined with any CCA-secure ES as DEM

  yields a CCA-secure PKES if the CDH is hard w.r.t. used groups.

- BUT: The ROM is only a heuristic for obtaining more efficient cryptographic constructions.

- Obviously the actual security crucially depends on the choice of the hash function h.

  - E.g.: $h(x) := 0^n$ for all x would yield a bad KDF.

  - In the ROM the concrete hash function is replaced by a random oracle, so this is completely left aside.

▷ Even worse, proofs in the ROM have no real implications:

[15] describes constructions which can be proven secure in the ROM, but which are insecure for any concrete hash function.

The RSA-PKES: Basic version

- **Definition**: Given $\mathsf{Gen}\mathbb{P}^2$:

  GenRSA: on input 1^n, run $\mathsf{Gen}\mathbb{P}^2(1^n)$ to obtain (p, q), compute $N = pq$, and $\lambda = \mathsf{lcm}(p-1, q-1)$; choose any $e \in \mathbb{Z}_\lambda^* - \{1\}$ and compute $d \in \mathbb{Z}_\lambda^*$ with $ed \equiv 1 \pmod{\lambda}$.

  - En-/decryption key: $\mathsf{ek}_I = (N, e)$, $\mathsf{dk}_I = (p, q, d)$
  - Plain-/ciphertexts: $\mathcal{M}_{\mathsf{ek}_I} = \mathcal{C}_{\mathsf{ek}_I} = \mathbb{Z}_N^*$

  Enc: on input $\mathsf{ek} = (N, e)$ and $m \in \mathbb{Z}_N^*$, outputs $m^e \bmod N$.

  Dec: on input $\mathsf{dk} = (N, d)$ and $c \in \mathbb{Z}_N^*$, outputs $c^d \bmod N$.

- Necessary: Has to be a TDP w.r.t. GenRSA.

▷ But: Not CPA-secure!

The basic RSA-PKES is deterministic and stateless.

▷ Fix: randomized encryption via randomized padding (later).

▷ **Ex**: Works also for $m \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$, but you shouldn't make use of this.

The RSA-PKES: Basic version – Remarks

- It is conjectured that e can be arbitrarily chosen from $\mathbb{Z}_\lambda^*$:
  - But see [13] (1998): Boneh et al. give some indication that – under certain restrictions – breaking RSA with small e might be easier than factoring.
  - On the other hand [3] (2009) shows – again under certain restrictions – that breaking RSA given (N, e) is as hard as factoring N – no matter how e was chosen.

▷ In order to speed up encryption, e is usually chosen small or of the form $e = 2^l + 1$ s.t. x^e can be easily computed by repeated squaring.

- d should always be at least that large that it cannot be found by brute-force enumeration.

▷ Wiener attack for $d < \frac{1}{3}N^{1/4}$ (see e.g. [11]).

The RSA-PKES: Basic version – Remarks

▷ **Ex**: Consider the following hybrid PKES:

The DEM is AES-based and uses a key k of size 512 bits (ES+MAC).

The KEM uses the RSA problem and encrypts k as $(k^e \bmod N)$.

Assume N has at least 2048 bits, and that $e = 3$.

Show how to efficienlty recover k given the ciphertext.

RSA-based PKES and the CRT

- Recall the CRT states that for $\gcd(p, q) = 1$:

$$\mathbb{Z}_{pq} \cong \mathbb{Z}_p \times \mathbb{Z}_q \text{ and } \mathbb{Z}_{pq}^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

  by means of the (ring) isomorphism

$$h \colon \mathbb{Z}_{pq} \to \mathbb{Z}_p \times \mathbb{Z}_q \colon x \mapsto (x \bmod p, x \bmod q).$$

▷ Further note that for p prime, we have for all $x \in \mathbb{Z}_p = \mathbb{Z}_p^* \cup \{0\}$:

$x^k \equiv x^{k \bmod p-1} \pmod p$

▷ Hence: for all $x \in \mathbb{Z}_N$ (not only $\mathbb{Z}_N^*$):

$$h(x^d) = (x^{d \bmod (p-1)} \bmod p, x^{d \bmod (q-1)} \bmod q).$$

▷ **Ex**: If $1 = \gcd(d, \lambda(N))$, then $x \mapsto x^d$ defines also a permutation on both $\mathbb{Z}_p$ and $\mathbb{Z}_q$.

RSA-based PKES and the CRT

- Consequences:
- We can speed up decryption by a factor 4:

  Bob precomputes $(d \bmod p-1, d \bmod q-1)$ and α, β such that $1 = \mathsf{gcd}(p,q) = \alpha p + \beta q$.

  He then may compute $c^d \bmod pq$ by means of

$$\left((c^{d \bmod (p-1)} \bmod p)q\beta + (c^{d \bmod (q-1)} \bmod q)p\alpha\right) \bmod pq.$$

▷ Advantage: Exponentiation modulo $N = pq$ (cost: $|N|^3$) is replaced by two exponentiations modulo p resp. q (total cost: $2\left(\frac{|N|}{2}\right)^3$).

- Encryption and decryption also work for $m, c \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$.

  Ex: For $x \overset{u}{\in} \mathbb{Z}_N$, the prob. that $x \notin \mathbb{Z}_N^*$ is negligible, i.e. the hardness of the RSA-TDP does not change if we extend its domain to $\mathbb{Z}_N$.

The RSA-PKES: PKCS #1 v1.5

- PKCS = public-key cryptography standard by the RSA labs, see here.
- PKCS #1 v1.5: (Variable-length randomized padding)

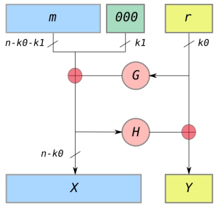
  Let a be the length of N in bytes, i.e., $a = \lceil \frac{|N|}{8} \rceil$.

  - Admissible plaintexts: $m \in (\{0,1\}^8)^*$ with $\frac{|m|}{8} \leq a - 11$.
  - Enc: given (N, e), choose $r \overset{u}{\in} (\{0,1\}^8)^{a-D-3}$, then msbf-interpret $x = 0^8||0^6 10||r||0^8||m$ as an integer in $\mathbb{Z}_N$, and output $x^e \bmod N$.

▷ Is assumed to be CPA-secure but it is not known whether hardness of the RSA problem is sufficient.

▷ It is not CCA-secure. See [10].

The RSA PKES: OAEP(+)



Taken from http://en.wikipedia.org/wiki/File:Oaep-diagram-20080305.png

- More recent RSA-variants (PKCS #1 v2.1) use the optimal asymmetric encryption padding (OAEP) by Bellare and Rogaway.
  - n is the bit length of N.
  - k_0, k_1 are predefined constants, see PKCS #1 v2.1.
    - Read $X||Y$ as msbf-representation of an integer $< N$.
  - G, H are hash functions/KDFs of sufficient output length.

The RSA PKES: OAEP($+$)

- At least in the ROM (treating G, H as ROs)

  RSA-OAEP($+$) is CCA-secure if the RSA problem is hard.

  - OAEP$+$ is a revised version proposed by Shoup [36].

  ▷ [36] gives a ROM-proof also for fixed encrytpion exponent $e = 3$.

▷ Recall: Proofs in the ROM do not imply security in the “real world”.

  - Deterministic functions used for instantiating the random oracles G, H still need to be chosen carefully.

  - [14] studies the question of what security properties hash functions need to satisfy in order to securely instantiate RSA-OAEP:

  ▷ **Ex**: $G(x) := 0^{n-k_0}$, $H(x) := 0^{k_0}$ is a bad choice under the assumption that $e = 3$ and k_0 is so small that $(0^{n-k_1}||r)^3 < N$:

  Eve can then distinguish $\left(\mathsf{OAEP}(0^{n-k_0-k_1})\right)^3 \bmod N$ from $\left(\mathsf{OAEP}(10^{n-k_0-k_1-1})\right)^3 \bmod N$.

RSA as KEM

- In practice, the used private-key ES will be built from, say, AES-128 and SHA-256 (block length 512), and, thus, the private key has fixed length 640.
- On the other hand, the size of the RSA modulus will grow over time:
  ECRYPTII (2012) recommends a 1776(+)bit modulus for 2017-2020
  BSI (2016) recommends a 3072(+) bit modulus for 2017-2021

▷ See www.keylength.com for a comparison of several recommendations.

▷ W/o randomized padding, we would need to require that the RSA problem is already hard for the negligible fraction of approx. $2^{640}/2^{3072}$ inputs.

  - Also recall the previous exercise when the key length of the DEM is too small w.r.t. the used encryption exponent.

▷ In fact, w/o randomized padding, a meet-in-the-middle attack allows to reduce the search space from 2^l to $2^{l/2}$ if l-bit keys are encrypted. (See [29] p.360.)

RSA as KEM

- Alternatively, we can proceed similar to how the DH protocol is used as KEM:

  Choose a random group element $x \in \mathbb{Z}_N^*$ and use KDF to derive from x the secret keys for the DEM.

  - Or we use it as a seed for the BM-construction based on the RSA-TDP to obtain a pseudorandom one-time pad to obtain a CPA-secure PKES.

    Bob simply uses the trapdoor to reverse the Blum-Micali construction and obtain the random seed used by Alice.

    This works for arbitrary TDPs. See the next slide.

▷ In combination with a CCA-secure DEM, the resulting PKES can be proven CCA-secure in the random oracle model, i.e., when the KDF is modeled as a random oracle. See [29], p.473.

Minimal assumptions for CPA-secure PKES

- Recall: Existence of CCA-secure ES, CPA-secure ES, comp. secret ES, PRGs, PRFs, PRPs, and OWFs are all equivalent.

▷ As every CPA-secure PKES can be used as a private-key ES, existence of OWFs is necessary also in the public-key setting.

- In contrast to the private-key setting, it is currently unknown if OWF suffice for CPA-secure PKES.

▷ Currently, it seems unlikely that OWFs suffice for CPA-secure PKES:

  - It is known that CPA-security is equivalent to the existence of trapdoor predicates [25].
  - By a result of [27] it seems unlikely that OWPs/OWFs imply trapdoor predicates.

- The existence of TDPs is sufficient for CPA-secure PKES to exist.

- **Definition**: Trapdoor PKES $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

  Let $(\mathsf{Gen}_f, \hbar, \mathsf{Smpl}, f)$ by a family of TDPs. Let hc be any hard-core predicate for this family.

  Gen: runs $\mathsf{Gen}_f(1^n)$ to obtain (I, td). Then $\mathsf{ek}_I := I$, $\mathsf{dk}_I := (I, \mathsf{td})$.

  Enc: given $m \in \{0,1\}^*$ of length l and ek, runs $\mathsf{Smpl}_{\mathsf{ek}}(1^n)$ to obtain $x \in \mathsf{Dom}_{\mathsf{ek}}$, then computes $\mathsf{Enc}_{\mathsf{ek}}(m) := (f^l_{\mathsf{ek}}(x), m \oplus \mathsf{BM}^l(x))$.

  Dec: given (y, c) and dk, sets $l := |c|$, computes $x := f^{-l}_{\mathsf{ek}}(y)$ using td, and finally outputs $m := c \oplus \mathsf{BM}^l(x)$.

- **Theorem**:

  The trapdoor PKES is CPA-secure relative to any family of trapdoor permutations.

Minimal assumptions for CCA-secure PKES

- So far, it seems that CCA-secure PKES require even stronger assumptions than CPA-secure PKES. [23]
  - In contrast to the private-key setting.
- But it can be shown that the existence of TDPs is also sufficient for CCA-secure PKES to exis [20].
- Is the existence of TDP necessary for CCA-secure PKES?
  - Probably not.
  - Already certain trapdoor one-way functions (TDF) suffice:

    Lossy TDFs resp. adaptive TDFs suffice. [31])

    ("TDF = OWF + trapdoor")
  - Lossy TDF are particular interesting:

    There exist problems which are conjectured to yield lossy TDFs and for which no efficient algorithm, not even for quantum computers is currently known – in contrast to factorization or discrete logarithm.

Minimal assumptions for secure PKES

- Trapdoor one-way function (TDF):

  Defined analogously to TDPs except that

  (i) the range and domain of f_J may differ, and

  (ii) f_J does not need to be injective.

  - Depending on the number of pre-images $\left|f_J^{-1}(y)\right|$:

    one-to-one ($\Rightarrow$) poly-to-one ($\Rightarrow$) super-poly-to-one

  ▷ (i) poly-to-one TDFs imply CPA-secure PKES;

  (ii) one-to-one TDF can be obtained from El Gamal/DDH;

  (iii) super-poly-to-one TDF can be obtained from OWFs, conjectured not to suffice for CPA-secure PKES. [8]

Minimal assumptions for secure PKES

$$\begin{tikzcd}
 & \text{spoly-to-one TDF} \arrow[r, leftrightarrow] & \text{OWF} & \\
 & \text{poly-to-one TDF} \arrow[u] \arrow[r] & \text{td. predicate} \arrow[r, leftrightarrow] & \text{CPA-sec. PKES} \arrow[ul] \\
 & \text{one-to-one TDF} \arrow[u] & \text{DDH hard} \arrow[l] \arrow[r] & \text{CCA-sec. PKES} \arrow[u] \\
\text{RSA hard} \arrow[r] & \text{TDP} \arrow[u] \arrow[urr] & &
\end{tikzcd}$$

- Arrows represent known implications of existence.
- See also here for further references.

Further public-key cryptosystems*

- Rabin (1979):

  CPA-security can be reduced to factoring the modulus $N = pq$.

  Allows to obtain a family of TDPs based equivalent to factoring.

- Goldwasser-Micali (1982):

  CPA-security can be reduced to deciding whether an $x \in \mathbb{Z}_N^*$ (with $N = pq$) is a square or not.

  (Trapdoor predicate)

- Paillier (1999):

  CPA-security con be reduced to deciding whether an $x \in \mathbb{Z}_{N^2}^*$ (with $N = pq$) is an N-th power modulo N^2 or not.

- See [29] (chapter) 11 for details.

Digital signatures: Motivation

- MAC scheme:

  The shared secret k allows Alice and Bob to verify that a received message m has originated from one of the two by means of the supplied MAC-tag t.

  In particular, if Eve tries to manipulate m, the origin of the message changes from Alice&Bob to Eve.

▷ Downside:

  Alice resp. Bob cannot prove to a third party that a given message (m, t) has been created explicitly by Bob resp. Alice.

- Possible fix:

  Make the secret k of how to "sign" a message m, i.e., compute the tag t, exclusive to Alice (Bob), and give everyone the information on how to verify that – except for negligible prob. – the tag t was indeed created by Alice for the received message.

Digital signature schemes: Definition

- **Definition**:

  digital signature scheme (DSS): PPT-algorithms $(\mathsf{Gen}, \mathsf{Sgn}, \mathsf{Vrf})$

  Gen: on input 1^n, outputs key parameters I describing the public verification key vk_I and the private signing key sk_I.

  - We assume that $|\mathsf{vk}_I|, |\mathsf{sk}_I| \geq n$ and that from both keys we can determine the set of the of admissible plaintexts/signatures ($\mathcal{M}_{\mathsf{sk}_I}$,$\mathcal{C}_{\mathsf{vk}_I}$).

  Sgn: on input a signing key sk and an admissible plaintext m, outputs a signature $\sigma \overset{r}{:=} \mathsf{Sgn}_{\mathsf{sk}}(m)$.

  Vrf: on input a verification key vk, an admissible plaintext m, and an admissible signature σ, outputs a single bit $\mathsf{Vrf}_{\mathsf{vk}}(m, \sigma) \in \{0, 1\}$.

  - σ is valid for m iff $\mathsf{Vrf}_{\mathsf{vk}}(m, \sigma) = 1$.

  We require that Vrf is deterministic (DPT) and that

  $$\mathsf{Vrf}_{\mathsf{vk}}(m, \mathsf{Sgn}_{\mathsf{sk}}(m)) = 1 \text{ for all } I \overset{r}{:=} \mathsf{Gen}(1^n) \text{ and } m \in \mathcal{M}_{\mathsf{sk}_I}.$$

DSS: Security

- **Definition**:

  A DSS $(\mathsf{Gen}, \mathsf{Sgn}, \mathsf{Vrf})$ is secure if every PPT-attack $\mathcal{A}$ succeeds only with negligible prob. $\varepsilon_{\mathcal{A}}(n)$ in the following experiment for every n:

  1. Alice runs $\mathsf{Gen}(1^n)$ to obtain sk and vk.

     She passes vk directly to Eve, while only admitting her oracle access to $\mathsf{Sgn}_{\mathsf{sk}}$.

     The oracle remembers the queries made by Eve in some list $\mathcal{Q}$.

  2. Eve runs $\mathcal{A}^{\mathsf{Sgn}_{\mathsf{sk}}}(\mathsf{vk})$ to obtain a message m and a signature σ.

  ▷ Eve succeeds if (1) $\mathsf{Vrf}_{\mathsf{vk}}(m, \sigma) = 1$ and (2) $m \notin \mathcal{Q}$.

- The definition essentially copies the definition of secure MAC.

DSS: Security – Remarks

- Minimal assumption for secure DSSs:
  - As for ES, MACs, and PKES, OWF have to exist.
  - In contrast to PKES, existence of OWFs already suffices. [32, 33].
- As in the case of MACs, it is only required that PPT-Eve is not capable of forging a valid tag for any “fresh” message m.
  - I.e., Eve can replay a valid message-tag pair previously intercepted.
  - ▷ Include nonces into the message as in the case of MACs.
- As for PKES, the definition does not consider the setting where Eve tries to attack the distribution of vk. (No man-in-the-middle.)
  - I.e., Alice still has to solve the problem of how to transmit vk securely to Bob remains.
  - ▷ For this use a Certificates/public-key infrastructure (PKI).

Certificates and PKI

- Important use of DSS:

  secure distribution of public keys

  under the assumption that a single trusted party (certificate authority (CA)) exists:

- Tasks of the CA:
  - guarantee the secure distribution of its own public-key pair $\mathsf{ek}_{\mathsf{CA}}, \mathsf{vk}_{\mathsf{CA}}$,
  - bind an entity A to its public-key pair $(\mathsf{ek}_A, \mathsf{vk}_A)$ by signing a certificate using $\mathsf{sk}_{\mathsf{CA}}$, and
  - create trust in these certificates:

- Distribution of $\mathsf{ek}_{\mathsf{CA}}, \mathsf{vk}_{\mathsf{CA}}$ e.g. via installation packages of web browsers.

Certificates and PKI

- "Example":

▷ Bob has generated $(\mathsf{ek}_B, \mathsf{dk}_B, \mathsf{sk}_B, \mathsf{vk}_B)$.

▷ He then goes to CA, identifies himself (ID card), and gives his public-key pair $(\mathsf{ek}_B, \mathsf{vk}_B)$ to CA.

▷ CA generates a certificate $\mathsf{cert}_{\mathsf{CA}\to B} := (m, \mathsf{Sgn}_{\mathsf{sk}_{\mathsf{CA}}}(m))$ with

e.g., $m =$ "[User-ID] Bob's public key is $(\mathsf{ek}_B, \mathsf{vk}_B)$. [Cert-ID]".

▷ Later: Alice wants to talk to Bob,

▷ She first obtains from somewhere a (claimed) certificate $\mathsf{cert}_{\mathsf{CA}\to B}$.

▷ She then uses the trusted $\mathsf{vk}_{\mathsf{CA}}$ to validate $\mathsf{cert}_{\mathsf{CA}\to B}$.

▷ She also checks on CA's webpage that "Cert-ID" has not been revoked.

▷ If verification succeeds, Alice trusts the ek_B extracted from $\mathsf{cert}_{\mathsf{CA}\to B}$.

Certificates and PKI

- Extension: several CAs allow to:
  - pick the certificate from the CA you trust the most.
  - transfer trust:

    CA_1 issues an “extended” certificate $\mathsf{cert}_{\mathsf{CA}_1 \to \mathsf{CA}_2} = m || \mathsf{Sgn}_{\mathsf{sk}_{\mathsf{CA}_1}}(m)$

    where e.g. $m =$
    “CA_1 trusts certificates by CA_2 using the public keys $(\mathsf{ek}_{\mathsf{CA}_2}, \mathsf{vk}_{\mathsf{CA}_2})$”.

  ▷ Leads to certificate chains:

    If Alice trusts CA_1, and CA_1 trusts CA_2, then the certificate chain $\mathsf{cert}_{\mathsf{CA}_1 \to \mathsf{CA}_2}, \mathsf{cert}_{\mathsf{CA}_2 \to B}$ allows her to obtain a trusted copy of ek_B.

- Generalization: web of trust (or “everyone is a CA”)
  - Alice obtains $\mathsf{cert}_{U_1 \to B}, \ldots, \mathsf{cert}_{U_l \to B}$, and decides based on her experience how much she trusts each user U_i resp. vk_{U_i}, to obtain some accumulated trust in the obtained keys, and chooses the most trusted one.

Certificates and PKI

- Assuming
  - a secure DSS,
  - a single trustworthy CA,
  - and sk_A is only known to Alice,

  non-repudiation can be achieved:
  - Assume Alice sends Bob the signed message (m, t) with $t := \mathsf{Sgn}_{\mathsf{sk}_A}(m)$.
  - After Bob has received (m, t), he can convince any other party that (m, t) originate from Alice:

    He tells the other party to obtain $\mathsf{cert}_{\mathsf{CA} \to A}$, and verify the signature itself.

  ▷ Except for negligible probability, only the owner of sk_A can have created $\mathsf{Sgn}_{\mathsf{sk}_A}(m)$.

Certificates and PKI

- Some problems of PKI:

▷ If Eve somehow obtains Alice's secret sk_A without her noticing it, Eve can impersonate her.

▷ Possible solution: smart cards

  - All private keys are stored on a smart card, and never "leave" the smart card:

    All computations requiring the private keys are done by the smart card.

  ▷ See e.g. Common Access Card.

  ▷ To obtain the key, Eve has to have physical access to the smart card.

▷ An untrustworthy CA can trick Alice or Bob into using the wrong keys (man-in-the-middle).

▷ See also [22] and [here] for a discussion of problems associated with PKI.

▷ For more details on PKI in general see, e.g. [2].

Constructing practical DSSs from PKES?

- DSS as the inverse of PKES?

  Set $\mathsf{sk} := \mathsf{dk}$, $\mathsf{vk} := \mathsf{ek}$, and $\mathsf{Sgn}_{\mathsf{sk}}(m) := \mathsf{Dec}_{\mathsf{sk}}(m)$, $\mathsf{Vrf}_{\mathsf{vk}}(m,t) := (\mathsf{Enc}_{\mathsf{vk}}(m) = t ? 1 : 0)$.

- In general, this construction is simply not applicable:

  E.g., consider a CPA-secure PKES where Enc is randomized (Vrf is required to be deterministic).

- The more precise formulation of the underlying idea is to use a TDP:

▷ For instance RSA-TDP: $\mathsf{Sgn}_{(N,d)}(m) = m^d \bmod N$ for $m \in \mathbb{Z}_N^*$.

  Problem: RSA-TDP is an isomorphism on $\mathbb{Z}_N^*$

  Forging a tag for $m \in \mathbb{Z}_N^*$:

  Choose any $m_1 \in \mathbb{Z}_N^*$ and set $m_2 := m_1^{-1} m \bmod N$.

  Then $\mathsf{Sgn}_{(N,d)}(m_1) \cdot \mathsf{Sgn}_{(N,d)}(m_2) \bmod N = \mathsf{Sgn}_{(N,d)}(m)$.

Constructing practical DSSs from TDPs/OWPs

- As a heuristic for "destroying" the algebraic structure underlying problems like RSA-TDP or DLP-OWP, a hash function is applied first to the given message.
  - For TDPs this is called the hash-then-invert paradigm.

▷ In case of RSA-TDP: (PKCS #1 v1.5)

Let $h : \{0,1\}^* \to \{0,1\}^n$.

- For simplicity, assume that the output of h can be interpreted directly as an element in $\mathbb{Z}_N^*$.

Set $\mathsf{Sgn}_{(N,d)}(m) := h(m)^d \bmod N$ and $\mathsf{Vrf}_{(N,e)}(m,t) := (h(m) \equiv t^e \pmod N ?1 : 0)$.

- At least it seems quite unlikely that for h a reasonable hash function W $h(m_1)^d \cdot h(m_2)^d \equiv h(m_1 \cdot m_2 \bmod N)^d \pmod N$ still holds.
- This also lifts possible restrictions on the admissible plaintexts.
- Necessary condition for h: collision resistance.

Constructing practical DSSs from TDPs/OWPs

- At least in the ROM, this heuristic can be justified for TDPs [6]:

▷ Under the idealization that $h : \{0,1\}^* \to \mathbb{Z}_N^*$ is a random oracle,

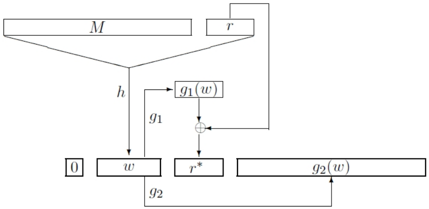
the security of the hash-then-invert paradigm based on RSA can be reduced to the hardness of the RSA problem.

▷ When replacing the RFO by a hash function in the "real world"

use a hash function/KDF with at least $\log_2 N$-bit output.

- This is called Full-domain-hash (FDH):
- Sidenote: if the same hash function is used to instantiate several RFOs, use distinct constants in the KDF constructions.
- Variant: Hash x into a stream of $\log_2 N$-bit numbers $(s_i)_{i\in\mathbb{N}}$ and take the first $s_i \in \mathbb{Z}_N^*$. [6]

RSA-PSS



Taken from [7].

- PKCS #1 v2.1 also includes the probabilistic signature scheme (PSS) construction by Bellare and Rogaway [7].
    - h is usually a standard hash function, while g a KDF.
    - Advantage of PSS compared to FDH+KDF:

      The actual message needs only to be hashed once.

    ▷ The signature is still a single element in $\mathbb{Z}_N$.
    - Security can be reduced to RSA-TDP in the ROM.

RSA-PSS

- **Definition**: PSS$[k, l]$

  For fixed security parameter n, let k, l be positive constants with $k + l < n$, let $h : \{0,1\}^* \to \{0,1\}^l$ and $g : \{0,1\}^l \to \{0,1\}^{n-l-1}$ be DPT-computable functions.

  Set $g_1(x)$ to be the first k bits of $g(x)$, and $g_2(x)$ the remaining $n - k - l - 1$ s.t. $g(x) = g_1(x)g_2(x)$.

  Gen: standard RSA-generator for (N, e, d) with $N \in [2^{n-1}, 2^n)$ and sk $:= (N, d)$, vk $:= (N, d)$.

  - Admissible messages: $\{0,1\}^*$

  Sgn: given $m \in \{0,1\}^*$, choose $r \overset{u}{\in} \{0,1\}^k$, compute $w = h(m||r)$, then $r^* = r \oplus g_1(w)$ and interpret $x = 0||w||r^*||g_2(w)$ as an integer in $\mathbb{Z}_N$ (msbf). Output Sgn$_{(N,d)}(m) \overset{r}{:=} x^d \bmod N$.

  Vrf: **Ex**.

- See PKCS #1 v2.1 for concrete values for k, l.

Elgamal's DSS

- Elgamal proposed in [21] also a stateful DSS:
  - Keys: Let p be a prime and g a generator of $\mathbb{Z}_p^*$.

    Choose $x \overset{u}{\in} \mathbb{Z}_p \setminus \{1, p-1\}$ and set $y := g^x \bmod p$.

    Then sk $= (p, g, x)$ and vk $:= (p, g, y)$.
  - Signing: Given $m \in \mathbb{Z}_p$, choose first $k \overset{u}{\in} \mathbb{Z}_{p-1}^*$; set $r := g^k \bmod p$; and compute $s := (m - x \cdot r) \cdot k^{-1} \bmod (p-1)$. Output (r, s).
  - Verification: Given (r, s), check that $g^m \equiv y^r r^s \pmod{p}$.
- Elgamal's DSS requires that the DLP is hard on $\mathbb{Z}_p^*$. But it is not known if this is already sufficient.
  - **Ex**: Recall when the DLP yields a collision-resistant CCF. Compare this CCF with Elgamal's DSS.
- [21] already notes that k must be kept secret and never used twice for signing a message for fixed private key sk $= (p, g, x)$. (See DSA later.)

ElGamal's DSS

- [21] also shows how to efficiently forge a valid tag for a new message:
  - Let (m, r, s) be a valid message-tag pair.
  - Choose $A, B, C \in \mathbb{Z}$ s.t. $\mathsf{gcd}(Ar - Cs, p-1) = 1$.
  - Set $r' := r^A \cdot g^B \cdot y^C \bmod p$, $s' := sr'(Ar - Cs)^{-1} \bmod p-1$, and $m' := r'(Am + Bs)(Ar - Cs)^{-1} \bmod p-1$.
  - **Ex**: Show that (r', s') is valid for m'.

▷ As a heuristic for preventing this attack, not the message m but instead its image $h(m)$ under a OWF $h : \{0,1\}^* \to \mathbb{Z}_{p-1}$ is signed.
  - I.e., the adversary needs to compute $h^{-1}(m')$ to use the above attack.
  - Obvioulsy, h needs also to be collision resistant.
  - Still, even when modeling h as random oracle, it is currently not known if the hardness of the DLP suffices.

- The digital signature algorithm is a variant of Elgamal's DSS working directly in a prime subgroup of $\mathbb{Z}_p^*$.

DSA

- **Definition**: DSA

  Let

  - $h_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ be a CRHF family, and
  - $\mathsf{Gen}\mathbb{G}_{\mathsf{DSA}}$ a group-generator s.t. $\mathsf{Gen}\mathbb{G}_{\mathsf{DSA}}(1^n)$ outputs (p,q,g) with $\langle g \rangle \leq \mathbb{Z}_p^*$ a cyclic group of prime order $q \geq 2^{l(n)}$.
  - For instance, use $\mathsf{Gen}\mathbb{QR}_{\mathsf{safe}}$.

  Gen: given 1^n,

  1. Run $\mathsf{Gen}\mathbb{G}_{\mathsf{DSA}}(1^n)$ to obtain (p,q,g).
  2. Choose $x \overset{u}{\in} \mathbb{Z}_q^*$.
  3. Compute $y := g^x \bmod p$.
  4. Output $\mathsf{sk} := (p,q,g,x)$ and $\mathsf{vk} := (p,q,g,y)$.

DSA

- **Definition**: DSA (cont'd)

  Sgn: given $m \in \{0,1\}^*$ and (p, q, g, x),

  1. Choose $k \overset{u}{\in} \mathbb{Z}_q^*$.
  2. Compute $R := g^k \bmod p$ and truncate it to $r := R \bmod q$.
  3. Let z be the integer with binary encoding $h_n(m)$.
  4. Compute $s = (z + x \cdot r) \cdot k^{-1} \bmod q$.
  5. If $r = 0 \vee s = 0$, go back to the first step, else return (r, s).

  Vrf: given $m \in \{0,1\}^*$ and $t = (r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q^*$,

  1. Let z be the integer encoding $h_n(m)$.
  2. Compute $u_1 = z \cdot s^{-1} \bmod q$ and $u_2 := r \cdot s^{-1} \bmod q$.
  3. Compute $R := g^{u_1} y^{u_2} \bmod p$.
  4. Output 1 iff $r \equiv R \bmod q$.

DSA

- Main differences to Elgamal's DSS:
  - Works directly in a subgroup $\langle g \rangle$ of $\mathbb{Z}_p^*$ with $q = |\langle g \rangle|$ prime.

    So generating elements in $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ is trivial.
  - Does not use $R = g^k \bmod p$, but its truncation $r := R \bmod q$:

    As $\langle g \rangle \cong \mathbb{Z}_q$, also yields a shorter signature, but Vrf needs to recover $g^k \bmod p$ in step (3) which requires $s \in \mathbb{Z}_q^*$.
- By adapting the truncation of the random group element R to an element of $r \in \mathbb{Z}_q$, the algorithm can also be generalized to elliptic curves (EC-DSA).
- The digital signature algorithm (DSA) is a DSS standardized by the NIST. See PKCS #1 v2.1 for a specification.
- As for Elgamal's DSS, it is not known if hardness of the DLP (or CDH/DDH) is also sufficient (also not in the ROM).

DSA

- Also for DSA the random k must be kept secret and must not be used twice:
- **Ex**: For given security parameter n let h be a fixed hash function, and $\mathsf{sk} = (p, q, g, x)$ and $\mathsf{vk} = (p, q, g, y)$.

  Now, let m_1, m_2 be two plaintexts with $h(m_1) \neq h(m_2)$. Assume that the same k is used for signing both messages s.t. $\mathsf{Sgn}_{\mathsf{sk}}(m_1) = (r, s_1)$ and $\mathsf{Sgn}_{\mathsf{sk}}(m_2) = (r, s_2)$. Further, assume that $r \neq 0$.

  Show how to obtain the signing key sk.

  - For Elgamal this works similarly, but becomes a bit more difficult as the group is not of prime order (no field).
  - See here, and here for an example of this attack.
  - Also when only a few bits of each k are revealed in each signature, the private key can be obtained, see, e.g., [9] or here.

CCA-security from CPA-security and secure DSS?

- Recall: CPA-secure ES + secure MAC yields CCA-secure ES via Enc-then-Mac.
- Analogous construction Enc-then-Sgn for PKES $(\mathsf{Gen}_{ES}, \mathsf{Enc}, \mathsf{Dec})$ and DSS $(\mathsf{Gen}_{DS}, \mathsf{Sgn}, \mathsf{Vrf})$:
  - $\mathsf{Gen}(1^n)$: runs $\mathsf{Gen}_{ES}(1^n)$ and $\mathsf{Gen}_{DS}(1^n)$ to obtain $\mathsf{ek}, \mathsf{dk}, \mathsf{sk}, \mathsf{vk}$.

    The public key is then $\mathsf{vek} = (\mathsf{vk}, \mathsf{ek})$, while the private key becomes $\mathsf{sdk} = (\mathsf{sk}, \mathsf{dk})$.
  - SgnEnc: given m, the sender uses his private sk_S, and receiver's public ek_R, to compute $\mathsf{Sgn}_{\mathsf{sk}_S}(\mathsf{Enc}_{\mathsf{ek}_R}(m))$.
  - DecVrf: given (m, t), the receiver uses his private dk_R and sender's public vk_S to compute $\mathsf{Dec}_{\mathsf{dk}_R}(c)$ only if $\mathsf{Vrf}_{\mathsf{vk}_S}(c, t) = 1$; otherwise $\perp$.
- An, Dodis, Rabin show in [4] that in general this construction does not yield a CCA-secure PKES, even if the underlying PKES is CCA-secure and the DS is secure.
  - But argue that this is mainly an artifact of the (too) strong notion of IND-CCA2.

Motivation

- **(Informal) reminder**: DLP is hard w.r.t. Gen$\mathbb{G}_{\mathsf{cyc}}$ if
  - Gen$\mathbb{G}_{\mathsf{cyc}}$: Given security parameter n outputs a description of a finite cyclic group $\langle \mathbb{G}, \cdot, 1 \rangle$.

    Description includes a generator g of $\mathbb{G}$, the size/order of $\mathbb{G}$, and an efficient algorithms for computing in $\mathbb{G}$.

    Description is polynomial in the security parameter n (e.g. not a table of the group operation!).
  - DLP instance: Given $y \in \mathbb{G}$ find any $x \in \mathbb{Z}$ s.t. $g^x = y$.
  - Hardness: Except for a negligible fraction of $y \in \mathbb{G}$ solving a DLP instance should take at least super-polynomial time in n.

Motivation

- **Examples**: DLP is conjectured to be hard w.r.t.
  - Gen$\mathbb{Z}^*_{\text{safe}}$: Description $(p, p-1, g)$ with $\langle g\rangle = \mathbb{Z}_p^*$ and $p \approx 2^n$ a safe prime. (Bad for DDH/El Gamal.)
  - Gen$\mathbb{QR}_{\text{safe}}$: Description: (p, q, g) with $\langle g\rangle = \mathbb{QR}_p$ and $p = 2q+1 \approx 2^n$ a safe prime.
- **Problem**: For these groups we know of superpolynomial but subexponential algorithms for solving the DLP:
  - index calculus algorithm (roughly: $O(2^{3n^{1/2}(\log_2 n)^{1/2}})$)
  - function field sieve (roughly: $O(2^{3n^{1/3}(\log_2 n)^{2/3}})$)
  - general number field sieve (roughly: $O(2^{3n^{1/3}(\log_2 n)^{2/3}})$)

▷ Hence, to achive a conjectured running time comparable to brute-forcing AES $n = \log|\mathbb{G}|$ should be at least (rough bound!)
  - 1500 for AES-128;
  - 7700 for AES-256.

Algorithms for solving the DLP

- **Possible fix**: Use groups for which we only know generic algorithms for solving the DLP:
  - Generic algorithms apply to any cyclic group but take in the worst-case time exponential in $\log|\mathbb{G}|$:
  - For instance: Baby-step giant-step algorithm, Pohlig-Hellman
  - As shown by Shoup [35]: For any generic algorithm there exist worst-case groups which make the algorithm run in time $\mathcal{O}(\sqrt{|\mathbb{G}|})$.
  - Recall: Every cyclic group $\mathbb{G}$ is isomorphic to $\langle\mathbb{Z}_{|\mathbb{G}|}, +, 0\rangle$ where the DLP is trivial to solve.

▷ Thus, to achieve a conjectured running time comparable to brute-forcing AES $n = \log|\mathbb{G}|$ should be at least (rough bound!)

- 256 for AES-128;
- 512 for AES-256.

▷ **Candidate** for such groups: certain elliptic curves.

Elliptic curves: Definition

- **Definition**: Let $\mathbb{F} = \langle \mathbb{F}, +, \cdot, 0, 1 \rangle$ be a field.

  The characteristic char$(\mathbb{F})$ is 0 if $k \cdot 1 = 1 + \ldots + 1 \neq 0$ for all k; otherwise it is the smallest such k.

- **Example**: char$(\mathbb{R}) = 0$, char$(\mathbb{Z}_p) = p$, char$(\mathrm{GF}(2^k)) = 2$.

- **Fact**: If char$(\mathbb{F}) > 0$, then it is prime.

- **Definition**: (char$(\mathbb{F}) \neq 2, 3$)

  Let $\mathbb{F}$ be a field with char$(\mathbb{F}) \neq 2, 3$, and $a, b \in \mathbb{F}$ with $\Delta_{a,b} = 4a^3 + 27b^2 \neq 0$. The (affine) elliptic curve described by $(\mathbb{F}, a, b)$ is

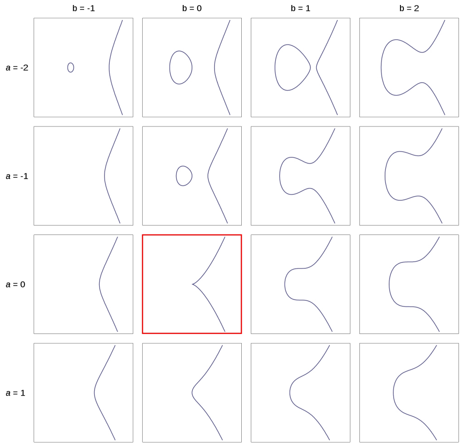
  $$\mathcal{E}(\mathbb{F}, a, b) := \{(x, y) \in \mathbb{F} \times \mathbb{F} \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

  with $\mathcal{O}$ some additional symbol (called the point at infinity).

  - For the cases char$(\mathbb{F}) = 2, 3$ see, e.g., [30, 37].

  ▷ Fields with char$(\mathbb{F}) = 2$ are also common in practice because of the natural binary representation.

Examples of elliptic curves over $\mathbb{F} = \mathbb{R}$



- $\{(x, y) \in [-3, 3]^2 \mid y^2 = x^3 + ax + b\}$ (Image taken from here).

▷ Note that $a = b = 0$ is by def. not a elliptic curve, as $\Delta_{a,b} = 0$.

▷ $\Delta_{a,b} \neq 0$ implies that $x^3 + ax + b$ has three distinct roots which guarantees that the curve is smooth, i.e. does not have any cusps.

Example of an elliptic curve over $\mathbb{F} = \langle \mathbb{Z}_{11}, +, \cdot \rangle$

- Note that $p = 11 \equiv 3 \pmod 4$, so we can compute a square root of a quadratic residue $z \in \mathbb{QR}_p$ by means of $z^{\frac{p+1}{4}} \bmod p$.
  - Recall: We can test if a $z \in \mathbb{Z}_p$ is a quadratic residue by computing the Legendre symbol $\left(\frac{z}{p}\right) = z^{\frac{p-1}{2}} \bmod p$.
- Let $a = b = 1$, i.e. the elliptic curve is defined by

$$y^2 \equiv x^3 + x + 1 \pmod{11}.$$

▷ So $\mathcal{E}(\mathbb{Z}_{11}, 1, 1)$ consists of the following points plus $\mathcal{O}$:

x	0	1	2	3	4	5	6	7	8	9	10
y^2	1	3	0	9	3	-1	3	-1	4	2	-1
y	± 1	± 5	0	± 3	± 5	$\perp$	± 5	$\perp$	± 2	$\perp$	$\perp$

Adding points on curves

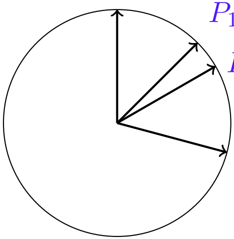
- How to add points on elliptic curves?
- Consider any curve in the plane given by means of a parametrization.

▷ For instance, a circle parametrized by $f(t) = (\sin t, \cos t)$:

$$P_0 := (\sin 0, \cos 0) = f(0)$$

$$P_1 := (\sin \tfrac{\pi}{4}, \cos \tfrac{\pi}{4}) = f(\tfrac{\pi}{4})$$

$$P_2 := (\sin \tfrac{\pi}{3}, \cos \tfrac{\pi}{3}) = f(\tfrac{\pi}{3})$$



$$P_1 \oplus P_2 := (\sin \tfrac{7\pi}{12}, \cos \tfrac{7\pi}{12}) = f(\tfrac{\pi}{4} + \tfrac{\pi}{3})$$

- Addition of points via addition of their parameters:

$$f(t_1) \oplus f(t_2) := f((t_1 + t_2) \bmod 2\pi)$$

Adding points on curves

- A parametrization of an elliptic curve

$$y^2 = x^3 + ax + b$$

over the reals is given by the Weierstraß's $\wp$-function.

- More precisely, the function $\wp$ is defined up to the choice of of constants, usually denoted by g_2, g_3, as the solution of a differential equation:

$$[\wp(z)']^2 = 4[\wp(z)]^3 - g_2\wp(z) - g_3$$

▷ Hence, for $g_2 := -a/4, g_3 := -b/4$ we obtain the parametrization

$$(x, y) := (\wp(z), 1/2 \cdot \wp(z)')$$

Adding points on curves

▷ Addition:

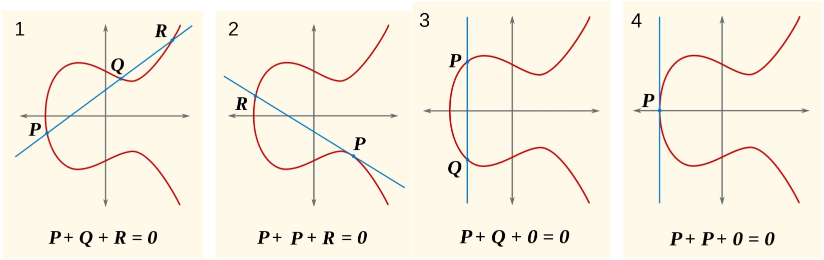
$$\begin{pmatrix} \wp(z_1) \\ 1/2 \cdot \wp'(z_1) \end{pmatrix} \oplus \begin{pmatrix} \wp(z_2) \\ 1/2 \cdot \wp'(z_2) \end{pmatrix} := \begin{pmatrix} \wp(z_1 + z_2) \\ 1/2 \cdot \wp'(z_1 + z_2) \end{pmatrix}$$

- Inverse:

$$-\begin{pmatrix} \wp(z_1) \\ 1/2 \cdot \wp'(z_1) \end{pmatrix} := \begin{pmatrix} \wp(-z_1) \\ 1/2 \cdot \wp'(-z_1) \end{pmatrix} = \begin{pmatrix} \wp(z_1) \\ -1/2 \cdot \wp'(z_1) \end{pmatrix}$$

▷ **Remark**: It can be shown that $\wp(-z) = \wp(z)$ and $\wp'(-z) = -\wp'(z)$.

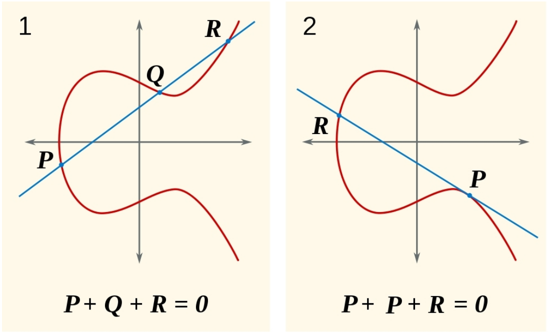
Adding points on curves



- In particular, it can be shown that $P, Q, R := -(P \oplus Q)$ are collinear.

▷ This can be used to define addition and inverse without knowing the parametrization.

Geometric interpretation for $\mathbb{F} = \mathbb{R}$: Finite case

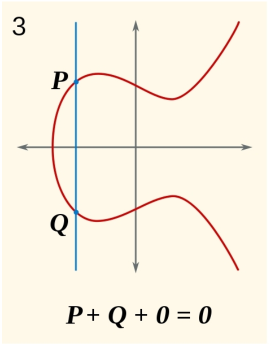


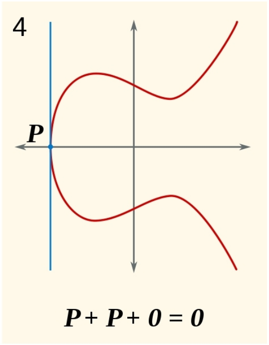
- The requirement $\Delta_{a,b} = 4a^3 + 27b^2 \neq 0$ guarantees that for $P, Q \in \mathcal{E}$ there is a unique point $R \in \mathcal{E} \setminus \{P, Q\}$ located on the line $\overline{PQ}$; if $P = Q$, let $\overline{PQ}$ be the tangent. Define $-R := (x_R, -y_R)$.

1 Case $x_P \neq x_Q$ with $P \oplus Q := -R$.

2 Case $P = Q \wedge y_P \neq 0$ with $P \oplus P := -R$ ("move Q into P").

Geometric interpretation for $\mathbb{F} = \mathbb{R}$: Infinite case





- If $\overline{PQ}$ does not intersect $\mathcal{E}$ in $\mathbb{F} \times \mathbb{F}$, take $\mathcal{O}$ (point at infinity).

3 Case $x_P = x_Q \wedge y_P = -y_Q$ with $P \oplus Q := \mathcal{O}$.

4 Case $x_P = x_Q \wedge y_P = y_Q = 0$ with $P \oplus Q := \mathcal{O}$.

- (Images taken from here.)

Elliptic curves: Group structure

- **Definition**: Let $\mathcal{E} = \mathcal{E}(\mathbb{F}, a, b)$ for $\mathsf{char}(\mathbb{F}) \neq 2, 3, \Delta_{a,b} \neq 0$.

  The group $\langle \mathcal{E}, \oplus, \mathcal{O} \rangle$ is defined as follows:

  Neutral: $\mathcal{O}$, i.e., $\mathcal{O} \oplus P = P \oplus \mathcal{O} = P$ for all $P \in \mathcal{E}$.

  Inverse: $-P := (x_P, -y_P)$ for $P = (x_P, y_P) \in \mathcal{E} \setminus \{\mathcal{O}\}$

  Addition: For $P, Q \in \mathcal{E} \setminus \{\mathcal{O}\}$ define $R := P \oplus Q$ as follows:

  - If $x_P \neq x_Q$, let $m := (y_Q - y_P) \cdot (x_Q - x_P)^{-1}$, or
    if $x_P = x_Q \wedge y_P = y_Q \neq 0$, let $m := (3x_P^2 + a)(2y_P)^{-1}$.

    Then $R = (x_R, y_R)$ with:

    $$x_R := m^2 - x_P - x_Q \text{ and } y_R := m(x_P - x_R) - y_P.$$

  - Else: $R := \mathcal{O}$.

- Projective coordinates, e.g. $x := \frac{u}{w}, y := \frac{v}{w}$, allow to reduce the number of inversion operations.

Elliptic curves: Group structure

- **Theorem**: (w/o proof)

  $\mathcal{E} = \langle \mathcal{E}(\mathbb{F}, a, b), \oplus, \mathcal{O} \rangle$ is a commutative group.

  If $q := |\mathbb{F}| < \infty$, then

  - Hasse's theorem:

    $|\mathcal{E}| \in [1 + q - 2\sqrt{q}, 1 + q + 2\sqrt{q}]$. (I.e.: $|\mathcal{E}| \approx |\mathbb{F}|$.)

  - Schoof-Elkies-Atkin (SEA) algorithm:

    $|\mathcal{E}|$ can be computed in DPT w.r.t. $\log q$.

Elliptic curves in cryptography

- For cryptograpy, usually not the whole curve is used, but only a sufficiently large cyclic subgroup $\langle P \rangle$ of prime order.
- Generic approach:
  - Pick a point P at random from the curve.
  - Check if its order is large enough and a prime:

    If the factorization of $|\mathcal{E}|$ is known, use the generator test.

    Complex multiplication appraoch [5]: Allows to generate an elliptic curve resp. its parameters a, b s.t. $|\mathcal{E}(\mathsf{GF}(q), a, b)| = S$ for given q, S.

Elliptic curves in cryptography

- Hasse's theorem ("$|\mathcal{E}| \approx |\mathbb{F}|$") guarantees that one can select a $P \in \mathcal{E}$ efficiently at random:
    - For $(x, y), (x, y') \in \mathcal{E}$ we have $y' = \pm y$.
    - ▷ So, at most two points on $\mathcal{E}$ can have the same x-coordinate.
    - ▷ At least $\frac{|\mathcal{E}|}{2} \approx \frac{|\mathbb{F}|}{2}$ distinct x-values in $\mathcal{E}$.
    - ▷ Hence, $x_P \overset{u}{\in} \mathbb{F}$ defines a point on $\mathcal{E}$ with prob. $1/2$.
    - ▷ Then solve $y^2 = x^3 + ax + b$ for y_P.
    - ▷ Recall: Computing a square root of a quadratic residue modulo a safe prime p (or any prime with $p \equiv_4 3$) is particularly easy:

      Simply compute $z^{\frac{p+1}{4}} \bmod p$.

Elliptic curves in cryptography

- [16] In practice, first a sufficiently large field $\mathbb{F}$ is chosen,
  - Most of the time, either $\mathbb{F} = \mathbb{Z}_p$ with $p \approx 2^n$ prime, or $\mathbb{F} = \mathsf{GF}(2^m)$.

  then $a, b \overset{u}{\in} \mathbb{F}$ are chosen, and $|\mathcal{E}|$ is determined for $\mathcal{E} = \mathcal{E}(\mathbb{F}, a, b)$.

▷ Next, one checks that $N := \frac{|\mathcal{E}|}{h}$ is prime for some sufficiently small h.

  - h is called the co-factor of the group, often $h \leq 4$. (cf. strong primes.)

  ▷ Allows to efficiently find a base point P of prime order N.

▷ Finally, $|\mathcal{E}|$ resp. N should satisfy:

  - $|\mathcal{E}| \neq |\mathbb{F}|$.
  - $p^B \not\equiv 1 \pmod{N}$ for all $1 \leq B \leq 100$[1].
  - Both $N - 1$ and $N + 1$ should have a large prime factor.

[1]Resp. $2^B \not\equiv 1 \pmod{N}$ for $1 \leq B \leq 100m$.

Elliptic curves in cryptography

- Regarding the restrictions on $|\mathcal{E}|$ and N:
  - See, e.g., appendix B of [16] for details and a discussion of attacks known in 2009.
  - Several types of curves considered sufficiently secure several years ago aren't used anymore today, so check up-to-date sources.
- The standard description of $\langle P \rangle \leq \mathcal{E}(\mathbb{F}, a, b)$ is then:
  - (p, a, b, P, N, h) for $\mathbb{F} = \mathbb{Z}_p$
  - $(m, f(x), a, b, P, N, h)$ for $\mathbb{F} = \mathbb{Z}_2[X]/f(x) \cong \mathsf{GF}(2^m)$
  - In practice, often precomputed descriptions are used. See, e.g., [17].

    Verify the correctness of precomputed descriptions!

Some problems:

- Implementation is error prone e.g.:
  - Missing check if point is indeed on the curve.
  - Faulty implementation of the group operation.
- Group operation is problematic w.r.t. side-channel attacks:
  - Three different cases when computing $P \oplus Q$.
  - ▷ Fix: A large fraction of elliptic curves can also be written as
    - Montgomery curve: $by^2 = x^3 + ax^2 + x$ ($b(a-4)^2 \neq 0$); or
    - Edwards curve: $x^2 + y^2 = 1 + dx^2y^2$ ($d \neq 0, 1, 2 \neq 0$) (cf. circle).

    which allows to speed-up addition and also make it more resilient to side-channel attacks.
  - ▷ But many NIST curves cannot be represented in this way.
- See the presentation by Bernstein and Lange [here] for more details and references.

Why “elliptic”?

- When determining the arc length of planar curves often so called elliptic integrals arise

$$\int R(x, \sqrt{P(x)})dx$$

with $R(x, y)$ a rational function and $P(x)$ a polynomial of degree $3, 4$.

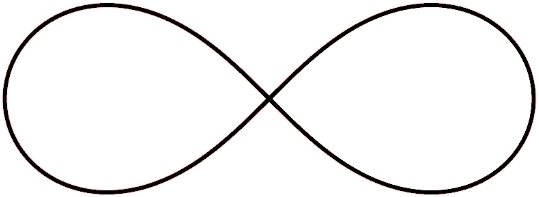
▷ For instance, the arc length of the ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ is given by

$$4a \int_{x=0}^{1} \sqrt{\frac{1 - k^2x^2}{1 - x^2}}dx \text{ with } k^2 = \frac{a^2 - b^2}{a^2} < 1$$

for

$$R(x, y) = \frac{1 - k^2x^2}{y} \text{ and } P(x) = (1 - k^2x^2)(1 - x^2).$$

Why "elliptic"?



Taken from here.

- Similarly, the arc length of the lemniscate $(x^2+y^2)^2 = a^2(x^2-y^2)$ is

$$\sqrt{8}a\int_{x=0}^{1}\frac{1}{\sqrt{1-x^4}}dx \text{ with } R(x,y)=\frac{1}{y}, P(x)=1-x^4.$$

- In general, such elliptic integrals cannot be expressed using "elementary" functions like rational functions, $\sqrt{x}, e^x, \sin x, \ldots$,

▷ and $E(t)=\int_0^t R(x,\sqrt{P(x)})dx$ can therefore be considered to define an independent class of functions.

Why “elliptic”?

- Around 1720, Fagnano showed a “doubling theorem” for the arc length of the lemniscate:

$$2\int_0^t \frac{dx}{\sqrt{1-x^4}} = \int_0^{\frac{2t\sqrt{1-t^4}}{1+t^4}} \frac{dx}{\sqrt{1-x^4}}.$$

- This result was extended by Euler to a general addition theorem for elliptic integrals.
- The inverse $E^{-1}(t)$ of an elliptic integral is called elliptic function.
- Abel and Jacobi were able to simplify and unify many of the results on elliptic integrals known so far by studying their respective elliptic functions.
- Later on Weierstraß showed that every elliptic function can be expressed as a rational function in two elliptic functions, the Weierstaß $\wp$-function, and its derivative $\wp'$.

[1] M. Abdalla, M. Bellare, and P. Rogaway. *DHIES: An encryption scheme based on the Diffie-Hellman Problem*. URL: http://www.cs.ucdavis.edu/~rogaway/papers/dhies.pdf.

[2] C. Adams and S. Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*.

[3] D. Aggarwal and U. Maurer. *Breaking RSA Generically is Equivalent to Factoring*. URL: http://eprint.iacr.org/2008/260.pdf.

[4] J. An, Y. Dodis, and T. Rabin. *On the security of joint signature and encryption*. URL: http://eprint.iacr.org/2002/046.pdf.

[5] H. Baier. *Efficient Algorithms for Generating Elliptic Curves over Finite Fields Suitable for Use in Cryptography*. URL: http://tuprints.ulb.tu-darmstadt.de/211/1/dissertation_harald_baier.pdf.

[6] M. Bellare and P. Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*. URL: http://cseweb.ucsd.edu/users/mihir/papers/ro.pdf.

[7] M. Bellare and P. Rogaway. *The exact security of digital signatures: How to sign with RSA and Rabin*. URL: http://cseweb.ucsd.edu/users/mihir/papers/exactsigs.pdf.

[8] M. Bellare et al. *Many-to-one trapdoor functions and their relation to public-key cryptosystems*. URL: http://cseweb.ucsd.edu/users/mihir/papers/tf.pdf.

[9] I. Blake and T. Garefalakis. *On the Security of the Digital Signature Algorithm*. URL: http://web-server.math.uoc.gr:1080/Members/theo/documents/documents/final.pdf.

[10] D. Bleichenbacher. *Chosen Ciphertext Attacks Against Protocolls Based on the RSA Encryption Standard PKCS # 1*. URL: http://www.springerlink.com/index/j5758n240017h867.pdf.

[11] J. Blömer and A. May. *A Generalized Wiener Attack on RSA*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.394&rep=rep1&type=pdf.

[12] D. Boneh. *The Decisional Diffie-Hellman Problem*. URL: http://crypto.stanford.edu/~dabo/pubs/papers/DDH.pdf.

[13] D. Boneh and R. Venkatesan. *Breaking RSA may be easier than factoring*.

[14] D. Brown. *What Hashes Make RSA-OAEP Secure?* URL: http://eprint.iacr.org/2006/223.pdf.

[15] R. Canetti and O. Goldreich. *The Random Oracle Methodology, Revisited*. URL: http://eprint.iacr.org/1998/011.pdf.

[16] Certicom. *Standards for Efficient Cryptography 1 (SEC 1)*. URL: http://www.secg.org/?action=secg,docs_secg.

[17] Certicom. *Standards for Efficient Cryptography 2 (SEC 2)*. URL: http://www.secg.org/?action=secg,docs_secg.

[18] R. Cramer and V. Shoup. *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*. URL: http://knot.kaist.ac.kr/seminar/archive/46/46.pdf.

[19] W. Diffie and M. Hellman. *New Directions in Cryptography*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.9720&rep=rep1&type=pdf.

[20] D. Dolev, C. Dwork, and M. Naor. *Non-malleable cryptography*.

[21] T. ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms*. URL: http://groups.csail.mit.edu/cis/crypto/classes/6.857/papers/elgamal.pdf.

[22] C. Ellison and B. Schneier. *Ten Risks of PKI: What You?e not Being Told about Public Key Infrastructure*. URL: http://www.schneier.com/paper-pki.pdf.

[23] Y. Gertner, T. Malkin, and S. Myers. *Towards a Separation of Semantic and CCA Security for Public Key Encryption*.

[24] O. Goldreich, Y. Lustig, and M. Naor. *On Chosen Ciphertext Security of Multiple Encryptions*. URL: http://eprint.iacr.org/2002/089.

[25] S. Goldwasser and S. Micali. *Probabilistic encryption*. URL: http://people.csail.mit.edu/joanne/shafi-pubs.html.

[26] D. Hofheinz and E. Kiltz. *Secure Hybrid Encryption from Weakened Key Encapsulation*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.5726&rep=rep1&type=pdf.

[27] R. Impagliazzo and S. Rudich. *Limits on the provable consequences of one-way permutations*.

[28] A. Joux and K. Nguyen. *Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups*.

[29] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*.

[30] A. Menezes. *Elliptic curve public key cryptosystems*. URL: http://books.google.com/books?id=bIb54ShKS68C&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false.

[31] D. Micciancio and O. Regev. *Lattice-based Cryptography*. URL: http://cseweb.ucsd.edu/users/daniele/papers/PostQuantum.pdf.

[32] M. Naor and M. Yung. *Universal one-way hash functions and the cryptographic applications*. URL: http://delivery.acm.org/10.1145/80000/73011/p33-naor.pdf?key1=73011%5C&key2=8646201921&coll=DL&dl=ACM&CFID=113052761&CFTOKEN=66962931.

[33] J. Rompel. *One-way functions are necessary and sufficient for secure signatures*. URL: http://www.cs.umd.edu/~jkatz/papers/rompel.pdf.

[34] V. Shoup. *A Proposal for an ISO Standard for Public Key Encryption (version 2.1)*. URL: http://www.shoup.net/papers/iso-2_1.pdf.

[35] V. Shoup. *Lower Bounds for Discrete Logarithms and Related Problems*. URL: http://www.shoup.net/papers/dlbounds1.pdf.

[36] V. Shoup. *OAEP Reconsidered*. URL: http://www.shoup.net/papers/oaep.pdf.

[37] A. Werner. *Elliptische Kurven in der Kryptographie*.