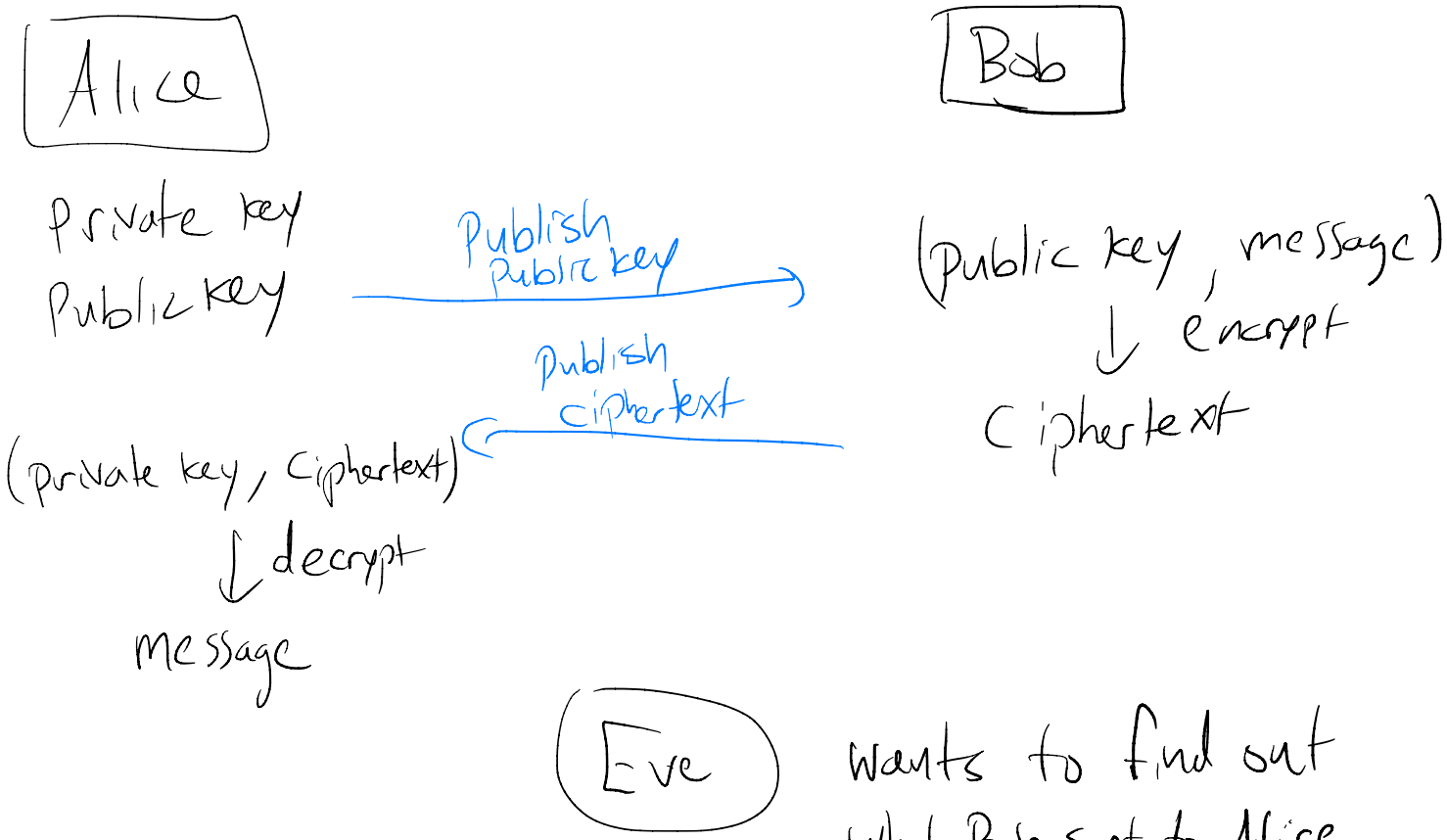
Lecture 17 Digital Signatures §4.1 & 4.2

Public key cryptosystem

For private communication from Bob to Alice



Eve wants to find out what Bob sent to Alice.

Digital Signature

Samantha wishes to sign a document so that Victor can verify it.

Samantha

Victor

private key = signing key

Public key = verifying key

Publish

Verifying key

(Document, Signing key)

↓ sign

Signature

Publish

(Document, Signature)

(Document, Signature, verifcation key)

↓ validate.

{true, false}

Faith

Wants to forge documents so Victor will think they come from Samantha.

Basically every important public key cryptosystem has an analogous Digital Signature scheme but some are more closely related than others.

The simplest to convert is RSA, which we just have to run backwards. The most commonly used is called DSA which is based on discrete log.

Examples

(1) RSA. Signatures.

Samantha

Victor

Signing key:
p, q primes
d exponent

Verification key:
$N = pq$, e s.t.
$de \equiv 1 \pmod{(p-1)(q-1)}$

(N, e) → Publish.

Signing

$D \in (\mathbb{Z}/N\mathbb{Z})^\times$

Compute $S \equiv D^d \pmod{N}$

(D, S) →

Verify: Compute
$S^e \equiv D \pmod{N}$.

Faith To forge a signature, need to find $S \equiv D^{Ye} \pmod{N}$. Same problem that Eve has to solve to break RSA encryption!

Rmk In practice, the document could be quite long. Instead we compute a cryptographic hash function $h : \{\text{documents of arbitrary length}\} \longrightarrow (\mathbb{Z}/N\mathbb{Z})^\times$

and then sign $h(D)$ instead of D itself.

A cryptographic hash function is a function w/ the property that given $y \in (\mathbb{Z}/N)^\times$ it is very hard to find D s.t. $h(D) = y$. This is important b/c if Samantha signs $(h(D)^d, D)$ and Faith can locate a document D' s.t. $h(D') = h(D)$, then Faith can convince Victor that Samantha has signed D'.

(This really happens too...)

As an aside, Bitcoin "proof of work" is about finding D s.t. $h(D) = y$ for a cryptographic hash function. This was designed to be hard & Bitcoin miners do it by brute force.

Examples Full disclosure, I have no clue how you would come up w/ these next two systems but DSA is in common use...

(2) Elgamal Digital signature

Samantha

Signing key: $a \in \mathbb{Z}/(p-1)\mathbb{Z}$

Verification key: $A = g^a \in \mathbb{F}_p^\times$ — Publish (p, A) → Victor

Choose random $k \in \mathbb{Z}/(p-1)\mathbb{Z}$

$D \in \{1, \dots, p-1\}$. ✱

$S_1 = g^k - p \cdot m \in \{1, \dots, p-1\}$

$S_2 = (D - aS_1)\, k^{-1} \in \mathbb{Z}/(p-1)$

Sign

Publish (D, S_1, S_2) →

Warning! We mix reduction mod p w/ reduction mod (p-1).

Verify

$$S_1^{S_2} A^{S_1} \equiv g^D \pmod{p}?$$

$$g^{ks_2} g^{as_1}$$

$$\| $$

$$g^{(D-as_1)} \cdot g^{as_1} = g^D$$

Faith

To forge a signature, must find S_1, S_2 such that

$$S_1^{S_2} A^{S_1} = g^D \overset{\log_g}{\iff} S_2 \log_g(S_1) + S_1 \log(A) = D$$

If Faith can solve discrete log problem, she picks random $S_1 = g^s$ & solves for S_2.

It is not known whether finding S_1 & S_2 is as hard as discrete log problem.

(3) Digital Signature Algorithm (DSA)

An efficiency improvement on Elgamal, and protects against Pohlig-Hellman attacks. In real life, the most common Signature scheme is Elliptic Curve DSA.

Pick a large prime p (the DSA specification says $p \approx 2^{1024}$) & a large prime q dividing $p-1$ (DSA spec says $q \approx 2^{160}$) Comparisons to ElGamal in blue.

Pick $g \in \mathbb{F}_p^\times$ of order q.

Samantha

Victor

Pick signing key

$a \in \mathbb{Z}/q$ (was $\mathbb{Z}/(p-1)$).

Verification key

$A = g^a \in (\mathbb{Z}/p)^\times$

Choose random

$k \in \mathbb{Z}/q\mathbb{Z}$

$D \in \mathbb{Z}/q\mathbb{Z}$ the document we are going to sign.

Compute $\leftarrow \{1, \ldots, p-1\}$

$$\begin{cases} s_1 = (g^k \bmod p) \bmod q \\ s_2 = (D + a s_1) k^{-1} \bmod q \end{cases}$$

Was $s_2 = (D - a s_1) k^{-1} \bmod p-1$

Send $\xrightarrow{(D, s_1, s_2)}$ Verify:

Check that

$$g^{D s_2^{-1}} A^{s_1 s_2^{-1}} \equiv s_1.$$

This works b/c:

$$\begin{aligned} g^{D s_2^{-1}} A^{s_1 s_2^{-1}} &\equiv g^{D s_2^{-1}} g^{a s_1 s_2^{-1}} \\ &\equiv g^{(D + a s_1) s_2^{-1}} \\ &\equiv g^k \end{aligned}$$

Apparently the hardness of forging a signature is the same as the hardess of breaking Diffie-Hellman.