RSA Encryption and Signatures

Lecture 17, CS 284, Autumn 2021

David Cash

Outline

1. Recall groups, $\mathbb{Z}_N^*$, public-key encryption
2. RSA Encryption
3. Digital Signatures: Motivation and Definitions
4. RSA Signatures

Corollary Let G be a finite abelian group of order $m > 1$. Let $e > 0$ be an integer relatively prime to m. Then the function f_e,

$$f_e : G \to G, \qquad f_e(g) = g^e$$
$$g \mapsto g^e$$

is a permutation. Moreover, if d is an inverse of e modulo m, then

$$f_d : G \to G \qquad f_d(g) = g^d$$
$$g \mapsto g^d$$

is the inverse of f_e.

For all g
$$f_d(f_e(g)) = g$$
$$= f_e(f_d(g))$$

The Group $\mathbb{Z}_N^*$ (Recall)

Intuition: Need to throw out not just $0 \in \mathbb{Z}_N$, but everything that does not have an modular inverse.

Def For a positive integer N, define

$$\mathbb{Z}_N^* = \{x \mid 1 \le x < N,\ \gcd(x, N) = 1\}.$$

Claim For each positive integer N, $\mathbb{Z}_N^*$ with operation $x \circ y = [xy \bmod N]$ is a group.

Proof Sketch: ID 1 ✓ INV x^{-1} is modular inverse of x mod N ASSOC: ✓

Euler's Theorem (!)

Theorem For any positive integer N, and integer a relatively prime to N,

$$a^{\varphi(N)} \equiv 1 \mod N.$$

Proof In $\mathbb{Z}_N^*$, $g^{|\mathbb{Z}_N^*|} = 1$ for any $g \in \mathbb{Z}_N^*$.

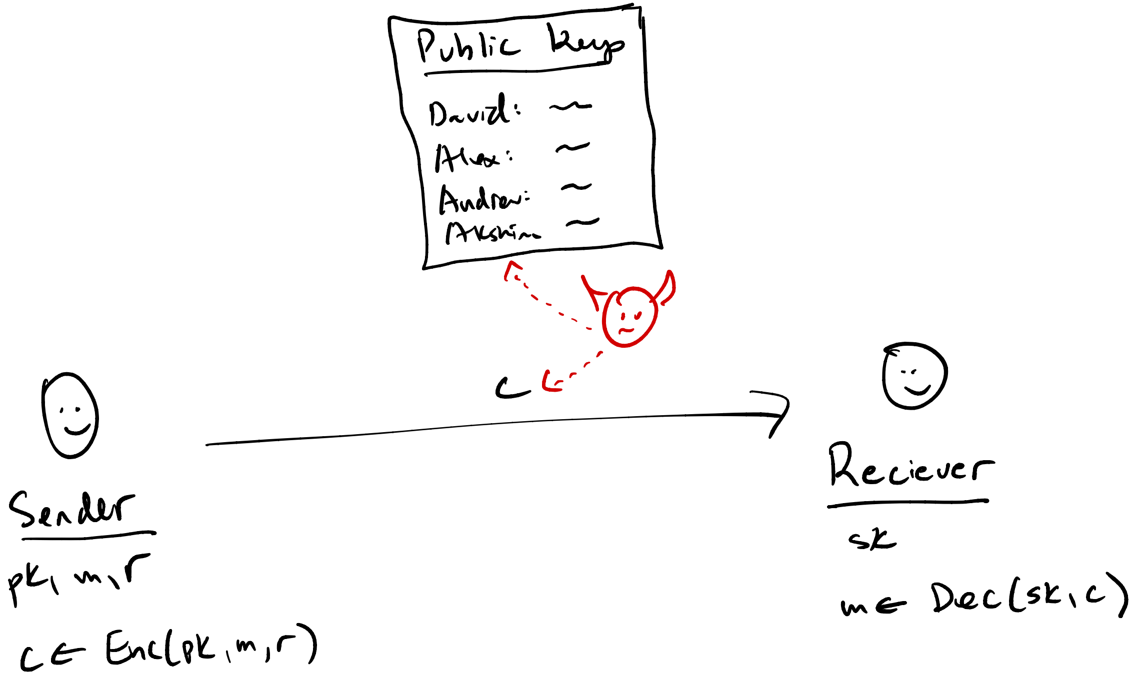
$\varphi(N)$

Public-Key Encryption Syntax

Def A public-key encryption scheme Π consists of three algorithms $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$, where

- KeyGen is randomized, takes no input besides random bits, and outputs two keys (pk, sk).
- Enc is randomized (with input r written explicitly), takes two more inputs pk, m, and outputs a ciphertext.
- Dec is deterministic, takes inputs sk, c and outputs m.

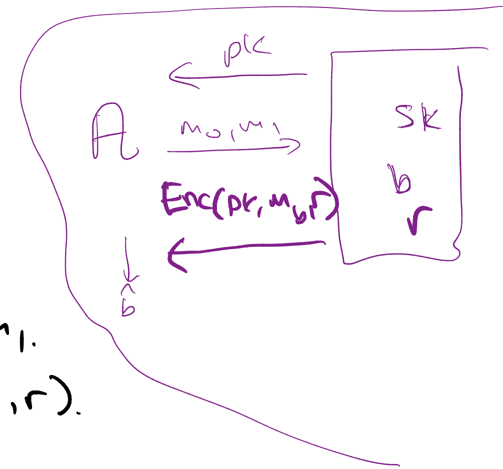
Chosen-Plaintext Attack Security: Motivation



- has pk and c, but not sk or r
- wants info about m; can influence what sender encrypts

Chosen-Plaintext Attack Security: Definition

Def Let $\Pi = (\text{Keygen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme, and let $\mathcal{A}$ be an adversary. Define $\text{Expt}_{\Pi}^{cpa}(\mathcal{A})$ by



<u>$\text{Expt}_{\Pi}^{cpa}(\mathcal{A})$</u>

1. Run $(pk, sk) \leftarrow \text{Keygen}()$
2. Give pk to $\mathcal{A}$. It chooses two messages m_0, m_1.
3. Pick $b \in \{0,1\}$, random r, compute $c \leftarrow \text{Enc}(pk, m_b, r)$.
4. Give c to $\mathcal{A}$. It outputs $\hat{b}$.
5. If $\hat{b} = b$ output 1. Else output 0.

Define $\text{Adv}_{\Pi}^{cpa}(\mathcal{A}) = \left| \Pr[\text{Expt}_{\Pi}^{cpa}(\mathcal{A}) = 1] - \frac{1}{2} \right|$.

Chosen-Plaintext Attack Security: Discussion

* No oracle for Enc; Just "one shot" for $\mathcal{A}$.
  ↳ But giving an oracle actually does not change definition much.

* Deterministic Enc algorithm ⇒ Can't have good CPA security

Outline

1. Recall groups, $\mathbb{Z}_N^*$, public-key encryption
2. RSA Encryption
3. Digital Signatures: Motivation and Definitions
4. RSA Signatures

The Main Idea: The RSA Trapdoor Function

RSA = Rivest, Shamir, Adelman

- Pick large primes $p \neq q$, set $N = pq$.
- Pick e relatively prime to $\varphi(N)$. (The order of $\mathbb{Z}_N^*$)
- Set $d = [e^{-1} \bmod \varphi(N)]$

$$\begin{tikzcd}
\mathbb{Z}_N^* \arrow[r, bend left, "{f_e(x) = [x^e \bmod N]}"] & \mathbb{Z}_N^* \arrow[l, bend left, "{f_d(y) = [y^d \bmod N]}"]
\end{tikzcd}$$

Use early "e/d" corollary, applied to the group $\mathbb{Z}_N^*$.

Basic (Deterministic) RSA Public-Key Encryption

"Textbook RSA"

Keygen: Pick k-bit primes $p \neq q$. $N \leftarrow pq$. Pick some small $e > 1$ such that $\gcd(e, \varphi(N)) = 1$. (know $\varphi(N) = (p-1)(q-1)$)

Set $d = [e^{-1} \bmod \varphi(N)]$.

Output $pk = (N, e)$, $sk = (N, d)$

MOD EXP

Enc(pk, m): Parse $pk = (N, e)$; Assume $m \in \mathbb{Z}_N^*$. Output $c = [m^e \bmod N]$.

Dec(sk, c): Parse $sk = (N, d)$. Output $m = [c^d \bmod N]$.

MOD EXP

Correctness? $f_d = f_e^{-1}$

	# bits
p, q	$k = 1024$
N	$2k \approx 2048$
e	2, 20...
d	$2k \approx 2048$

Questions to Address

1. Ok to assume $m \in \mathbb{Z}_N^*$? ✓

2. Is this secure?

3. This is deterministic. How can we randomize it?

Inverting RSA $\Longleftrightarrow$ Computing "e^{th} root mod N"

- $e = 3$ used to be common. $e = 65537$ is used now.

Inverting $Enc(pk, m)$ $\Longleftrightarrow$ finding m such that $[m^3 \bmod N] = c$.

* Without the "mod" this is easy. (find m such that $m^3 = c$).

Security Intuition for RSA

Given $pk = (N, e)$ and $c = [m^e \bmod N]$, can someone find m?

Hopeful thinking:

① To find m, need $d = [e^{-1} \bmod \varphi(N)]$

② To find d, need $\varphi(N) = (p-1)(q-1) = pq - p - q + 1$

③ To find $\varphi(N)$, need p and q.

* Finding p, q from $N = pq$ is the "factoring problem".

Factoring Algorithms (Details are optional info)

Algorithm	Time to factor N
Naive: Try dividing N by 1, 2, ...	$\sqrt{N} = \exp(0.5 \cdot \ln(N))$
Meet-in-the-Middle type attack	$N^{1/4} = \exp(0.25 \cdot \ln(N))$
"Quadratic Sieve"	$O(\exp(\ln(N)^{1/2} \cdot \ln(\ln(N))^{1/2})) \approx N^{\frac{1}{\ln(N)^{1/2}}}$
"Number Field Sieve"	$O(\exp(1.9\ \ln(N)^{1/3} \cdot \ln(\ln(N))^{2/3})) \approx N^{\frac{1}{\ln(N)^{1/3}}}$

Total break would be $\ln(N)^c = \exp(c \cdot \ln(\ln(N)))$

Factoring Records

Challenges posted by RSA Labs

Bit Length of N	Year Factored
400	1993
478	1994
515	1999
768	2009
795	2019
829	2020

Deterministic RSA Security Problems (Even if factoring is hard)

① $\mathrm{Enc}(pk, 1) = \underline{1}$

② If $m < N^{1/3}$, then $\mathrm{Enc}(pk, m) = m^3$ (as a real number)

③ Malleability: Given pk and $c = [m^e \bmod N]$ (but not m), can compute c' that decrypts to $2m$.

$c' = [2^e \cdot c \bmod N]$

$= [2^e \cdot m^e \bmod N]$

$= [(2m)^e \bmod N]$

Randomized RSA (Sketch)

During encryption, add random padding that is removed.

$$\mathrm{Enc}(pk, m, r) = [(r \| m)^e \bmod N]$$

concatenate message and randomness, treat result as element of $\mathbb{Z}_N^*$.

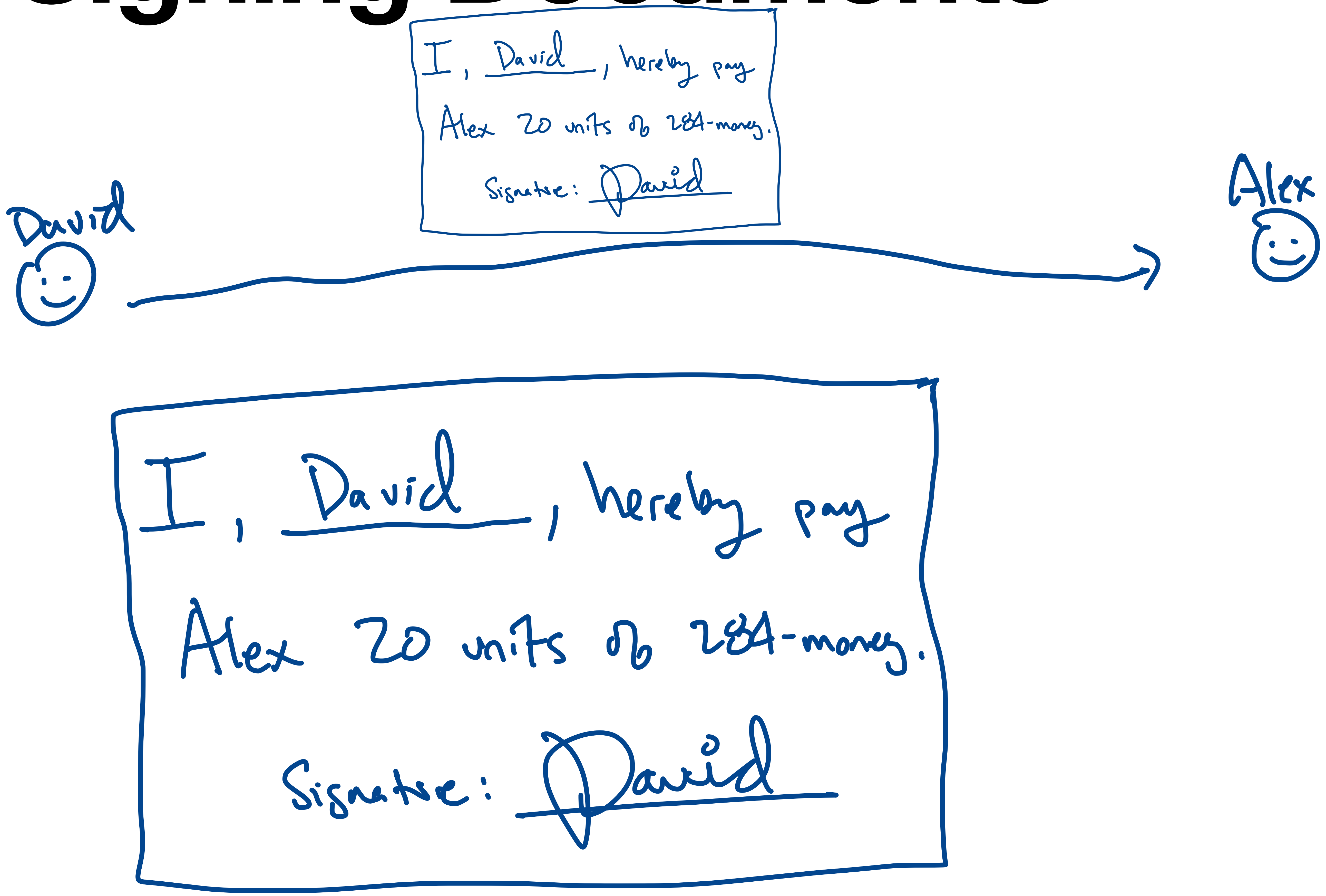
→ Malleability ???

Bleichenbacher's Padding Oracle

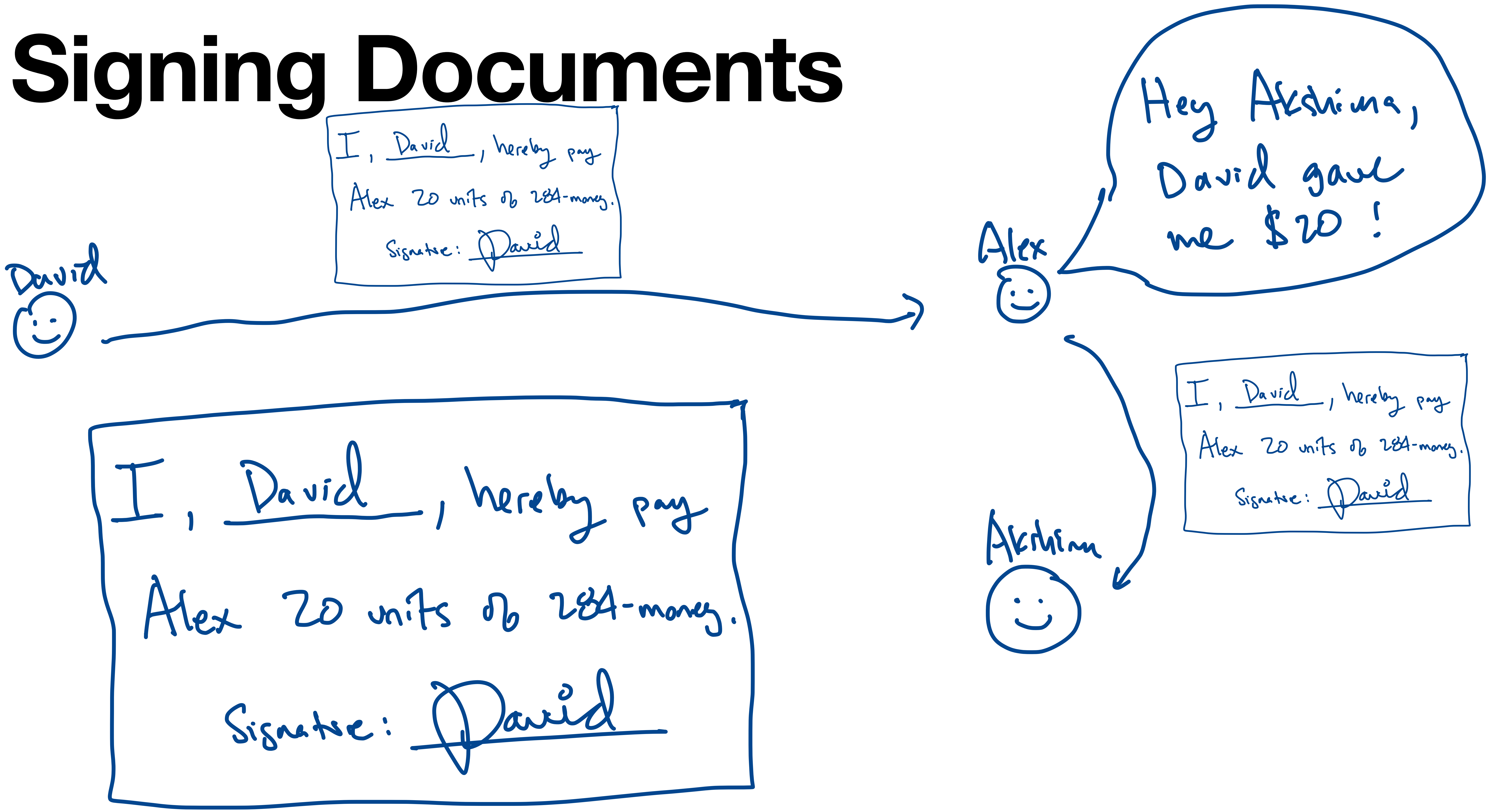
Outline

1. Recall groups, $\mathbb{Z}_N^*$, public-key encryption
2. RSA Encryption
3. Digital Signatures: Motivation and Definitions
4. RSA Signatures

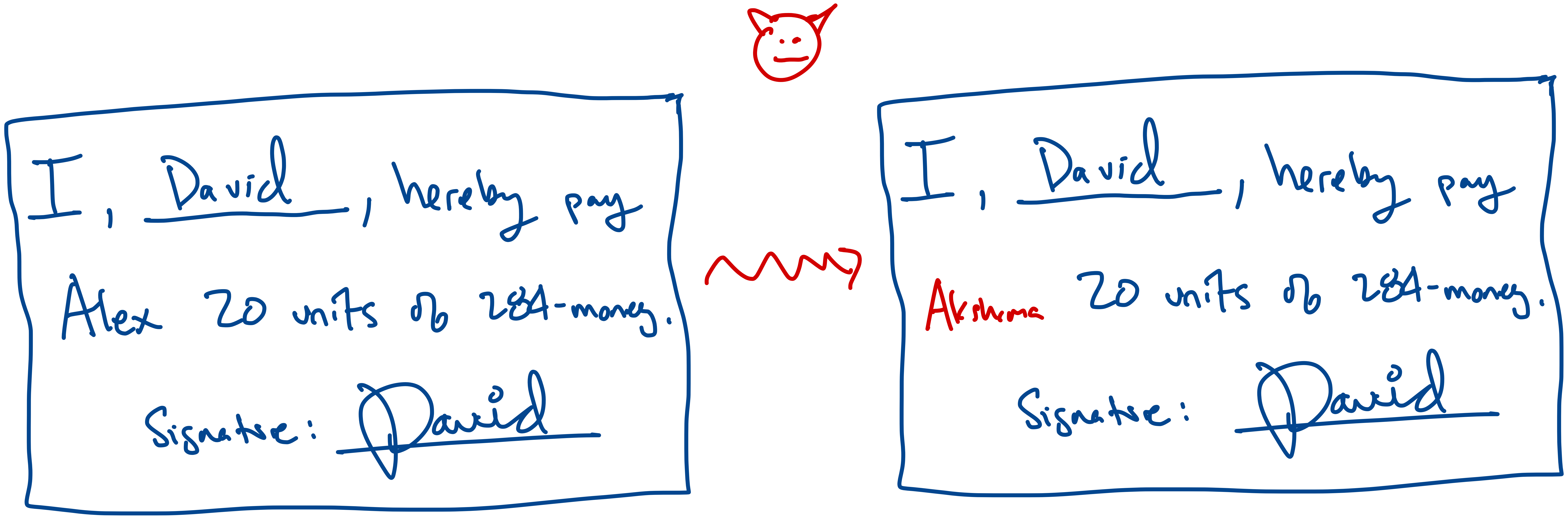
Signing Documents



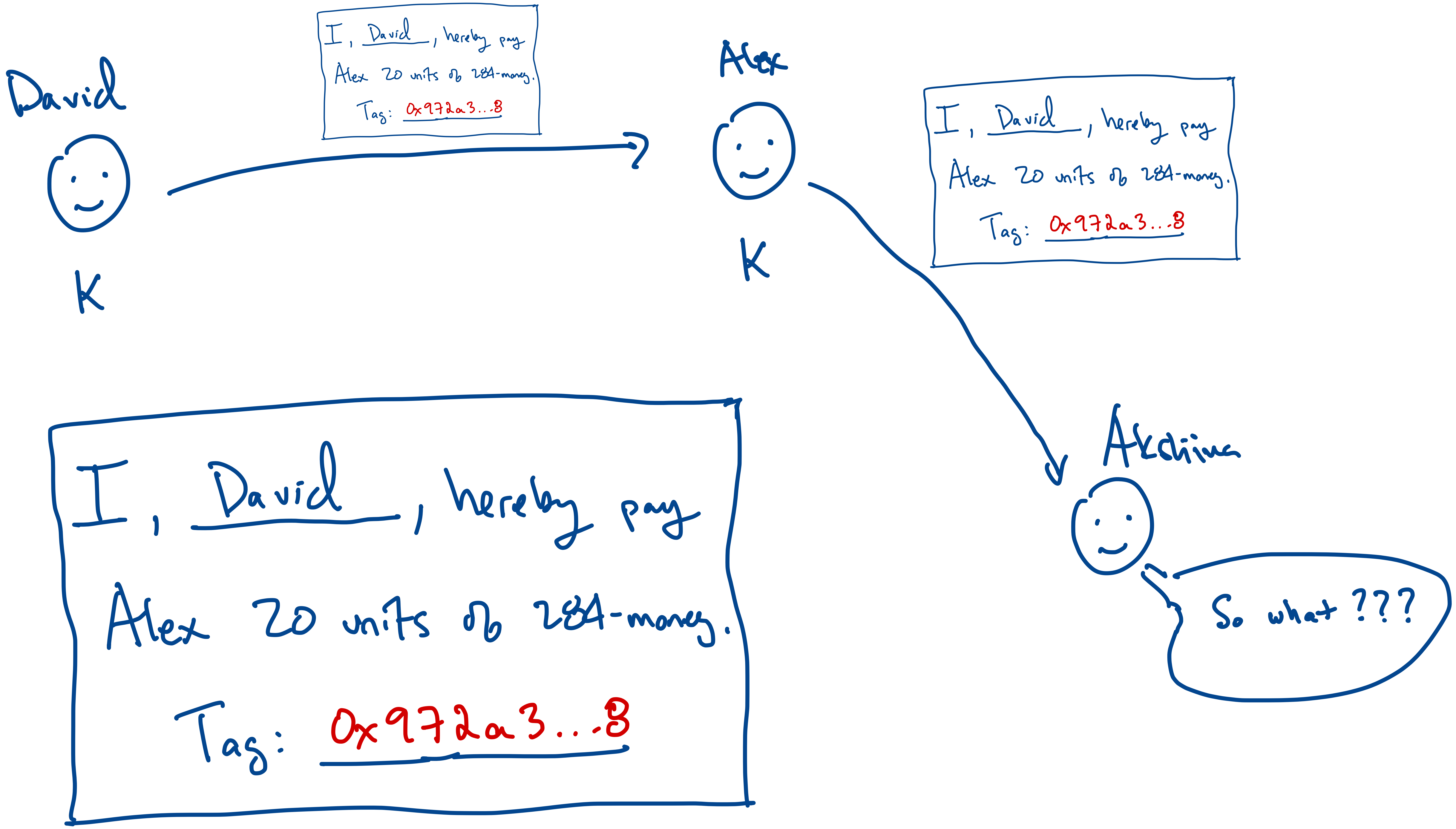
Signing Documents



Electronic Signatures?



What about using a MAC?

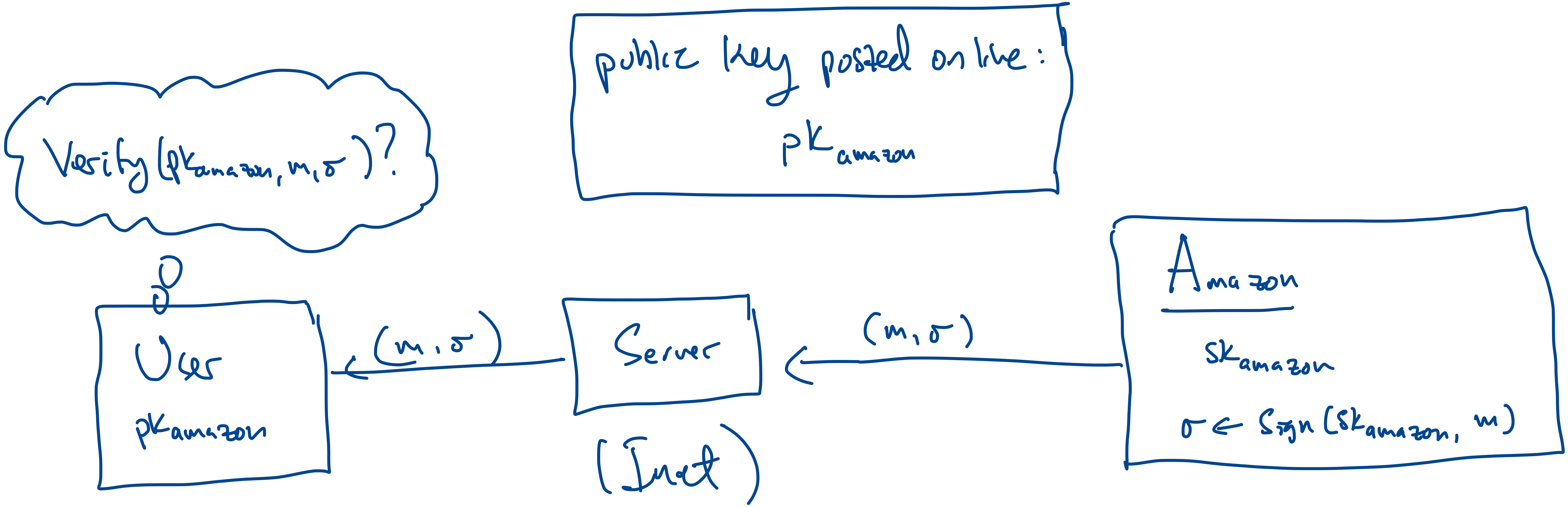


Digital Signatures: Syntax

<u>**Definition:**</u> A *digital signature scheme* with message space $\mathscr{M}$ consists of three algorithms $\Pi = (\text{KeyGen}, \text{Sign}, \text{Verify})$ with the following syntax:

- KeyGen takes no input, and outputs a pair of keys (pk, sk).
- Sign takes inputs sk and $m \in \mathscr{M}$ and outputs a “signature” σ
- Verify takes inputs pk, $m \in \mathscr{M}$, and σ and outputs a bit.

How to Use Digital Signatures

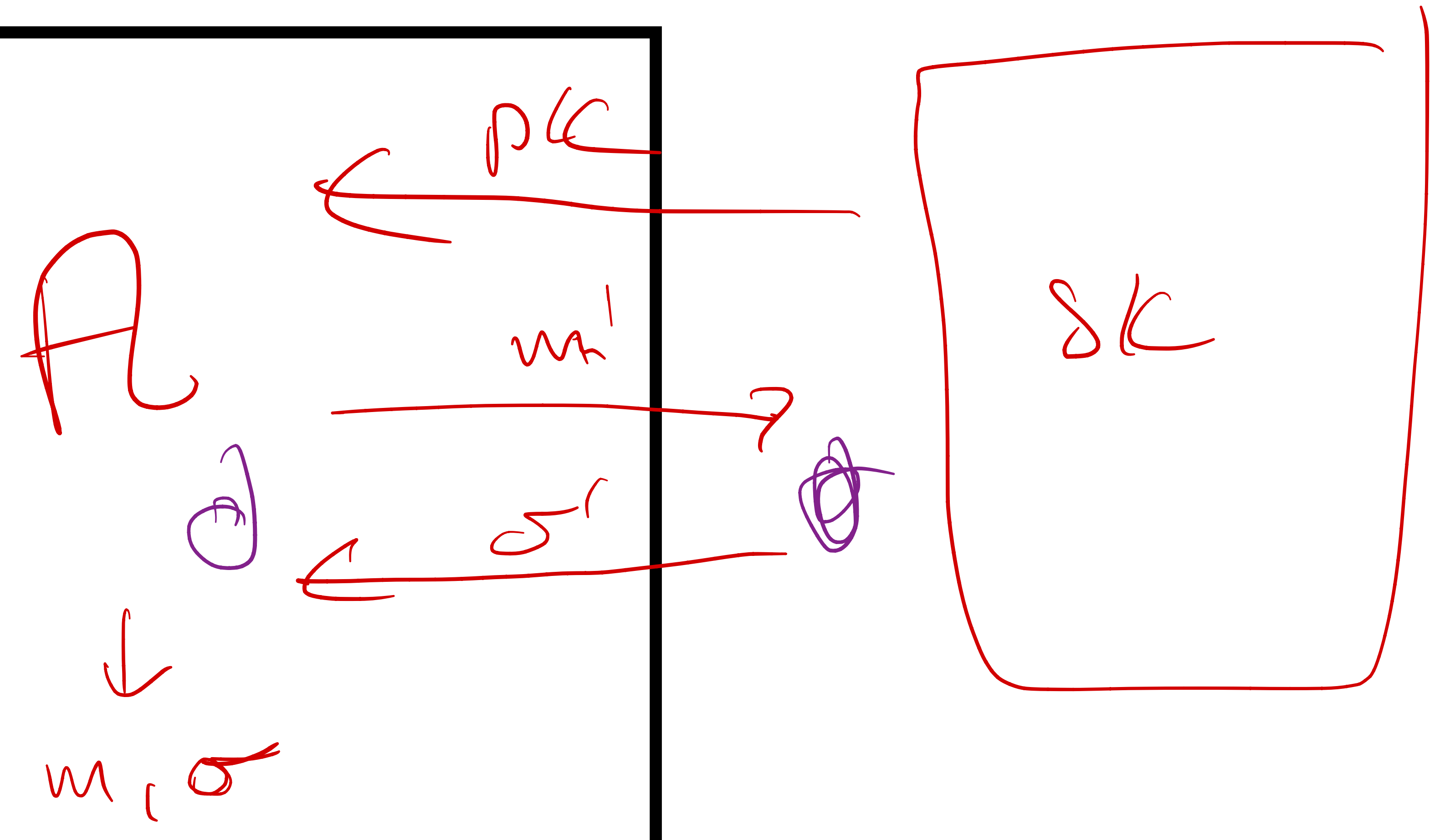


Security of Digital Signatures: Definition

Definition: Let $\Pi = (\text{KeyGen}, \text{Sign}, \text{Verify})$ be a digital signature scheme with message space $\mathscr{M}$ and let $\mathscr{A}$ be an adversary. Define

$\text{Expt}^{\text{uf}}_{\Pi}(\mathscr{A})$:

1. $(pk, sk) \leftarrow \text{KeyGen}$
2. Run $\mathscr{A}$ with input pk and oracle $\text{Sign}(sk, \cdot\,)$.
3. Eventually $\mathscr{A}$ outputs m, σ.
4. If $\text{Verify}(pk, m, \sigma) = 1$ and m was never queried to $\mathscr{A}$'s oracle then output 1.
5. Else output 0.

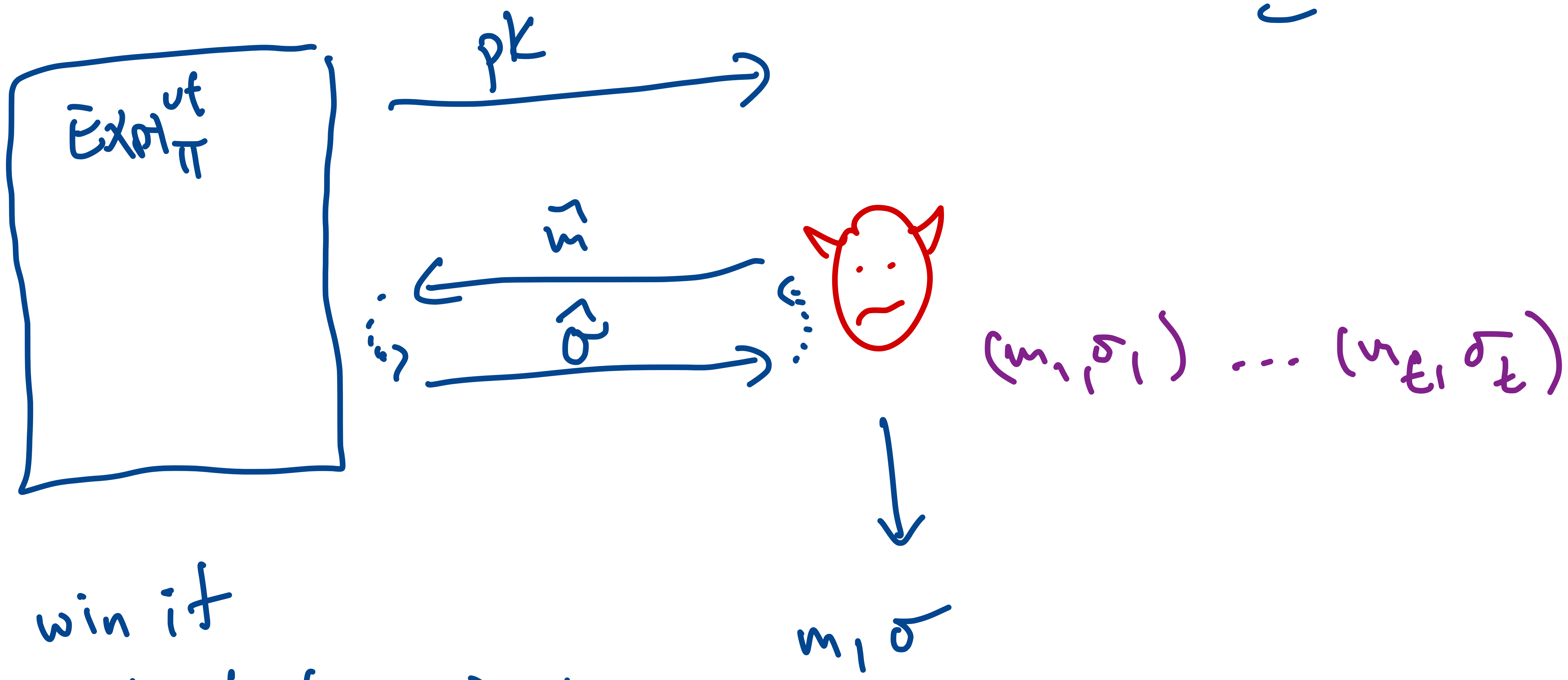


Finally, let $\text{Adv}^{\text{uf}}_{\Pi}(\mathscr{A}) = \Pr[\text{Expt}^{\text{uf}}_{\Pi}(\mathscr{A}) = 1]$.

Security of Digital Signatures: Idea

- Goal: Hard to forge, even after seeing pk + lots of signatures

Q: One try at forgery?



win if

- $Verify(pk, m, \sigma) = 1$
- m never queried

Outline

1. Recall groups, $\mathbb{Z}_N^*$, public-key encryption
2. RSA Encryption
3. Digital Signatures: Motivation and Definitions
4. RSA Signatures

"Textbook"

Plain RSA Signatures (Insecure!)

Fix a bit-size n for RSA primes and some exponent $e > \overset{1}{\cancel{0}}$. (e.g. $n \approx 1024$, $e = 65537$).
Message space $\mathcal{M} \subseteq \{0,1\}^{2n-1}$.

- KeyGen: Choose random n-bit primes p, q. Let $N \leftarrow pq$, $\varphi(N) = (p-1)(q-1)$, and $d \leftarrow [e^{-1} \bmod \varphi(N)]$. Let $pk \leftarrow (N, e)$, $sk \leftarrow (N, d)$. Output (pk, sk)
- $\text{Sign}(sk, m)$: Parse sk as (N, d) and view m as integer in $\mathbb{Z}_N^*$. Output

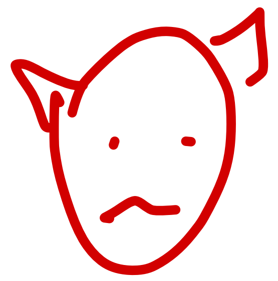
$$\sigma = [m^d \bmod N].$$

- $\text{Verify}(pk, m, \sigma)$: Parse pk as (N, e) and view m, σ as integers in $\mathbb{Z}_N^*$. Output 1 iff

$$m = [\sigma^e \bmod N].$$

Attack #1 on Plain RSA

Sign(sk, ·)



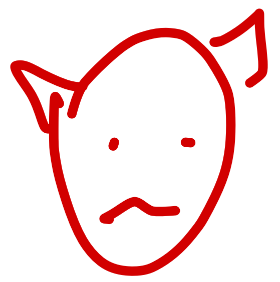
On input pk: Output $m = 1$, $\sigma = 1$.

→ Wins because

- Verify(pk, 1, 1) checks if $\sigma^e = [m \bmod N]$
- m never queried.

Attack #2 on Plain RSA : "Malleability"

$\mathrm{Sign}(sk, \cdot)$



On input pk:

1. Pick some $\hat{m}_1, \hat{m}_2 \in \mathbb{Z}_N^*$, $\hat{m}_1 \neq \hat{m}_2$, neither equal to 1.
2. Query $\hat{m}_1$ to get $\hat{\sigma}_1$.
3. Query $\hat{m}_2$ to get $\hat{\sigma}_2$.
4. Set $m = [\hat{m}_1 \cdot \hat{m}_2 \bmod N]$, $\sigma = [\hat{\sigma}_1 \cdot \hat{\sigma}_2 \bmod N]$.
5. Output (m, σ).

$$\sigma^e \equiv (\hat{\sigma}_1 \hat{\sigma}_2)^e \equiv (\hat{m}_1 \cdot \hat{m}_2)^{d \cdot e} \equiv \hat{m}_1 \cdot \hat{m}_2 \equiv m \bmod N$$

Attack #3 on Plain RSA: "Backwards Signing"

On input pk:

1. Pick $\sigma \leftarrow \mathbb{Z}_N^*$.
2. Set $m \leftarrow [\sigma^e \bmod N]$.
3. Output (m, σ).

Wins because:

- No queries.
- $[\sigma^e \bmod N] = m$ by definition!

Securing RSA Signatures with Hashing

- Use a hash function $\text{Hash}: \{0,1\}^* \to \{0,1\}^{2048}$
- KeyGen is unchanged

Sign(sk, m)

1. Parse sk as (N, d)
2. $x \leftarrow \text{Hash}(m)$
3. $\sigma \leftarrow [x^d \bmod N]$
4. Output σ.

Verify(pk, m, σ)

1. Parse pk as (N, e)
2. $x \leftarrow \text{Hash}(m)$
3. If $x \stackrel{?}{=} (\sigma^e \bmod N)$ output 1,
   Else output 0.

Security of RSA Signatures with Hashing

① Signature of $m=1$ is no longer $\sigma = 1$ $\left(1^e \neq H(1) \right)$

② Malleability is prevented $\left(H(m_1) \cdot H(m_2) \neq H(m_1 \cdot m_2) \right)$

③ Backwards Signing is prevented

↳ $\mathcal{A}$ picks σ, wants to find m such that $\sigma^e = H(m)$ ...

$\Rightarrow$ Hash should be preimage-resistant.

④ New threat: Collisions

↳ If $\mathcal{A}$ finds collision $m_1 \neq m_2$, $H(m_1) = H(m_2)$ then a signature on m_1 is also a signature on m_2.

Choosing the Hash Function for RSA Signatures

Using SHA256 (or similar) is insecure! (for subtle reasons)

Say N is 2048 bits long.

$$x = H(m) = \overset{2048}{\boxed{0 \cdots\cdots 0}}\overset{256}{\boxed{\text{SHA256}(m)}}\overset{0}{}$$

→ An algorithm called "index calculus" can find $[x^{1/e} \bmod N]$ relatively quickly.

→ Instead use a "full domain hash"

$$x = \text{Hash}(m) = \overset{2048}{\boxed{\text{SHA256}(1 \| m) \,\|\, \text{SHA256}(2 \| m) \,\|\, \cdots \,\|\, \text{SHA256}(t \| m)}}\overset{0}{}$$

The End