CSC3631 Cryptography
- Digital Signatures I

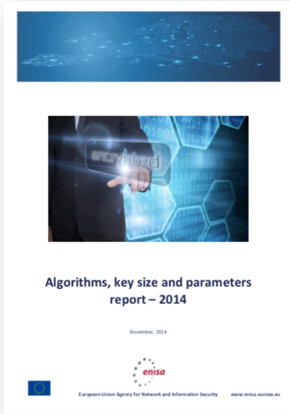
Thomas Gross

ENISA and NIST Key Size Recommendations

Symmetric Ciphers:
- ✓ **Block ciphers:** AES-256
- ✓ **Stream:** HC-128 & Snow 3G
- **Do not use RC4!**

Hashing:
- ✓ **Hash:** ≥ SHA-2 256

Asymmetric Encryption:
- ✓ **Encryption:** ≥ RSA-3072
- **Do not use RSA-1024!**



[ENISA Report Algorithms, Key Sizes and Parameters, Nov. 2014]

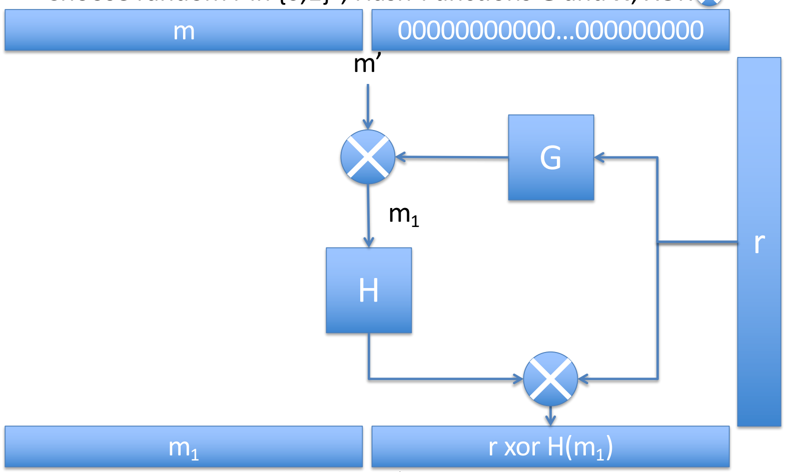
How do we use **RSA Encryption** in practice?

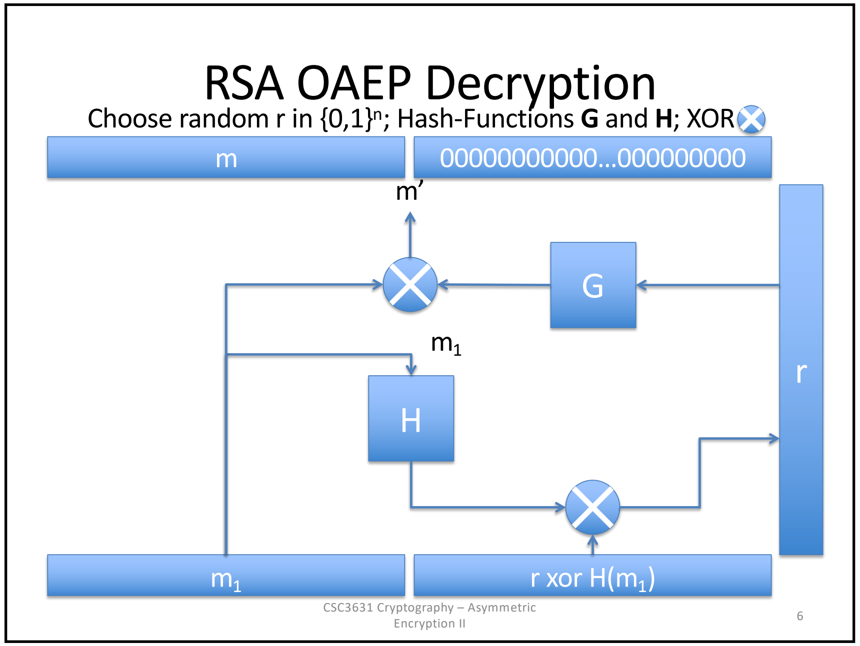
Use appropriate padding (PKCS/OAEP)
- Randomization
- Structure
- Use of full message length

Use Hybrid Encryption

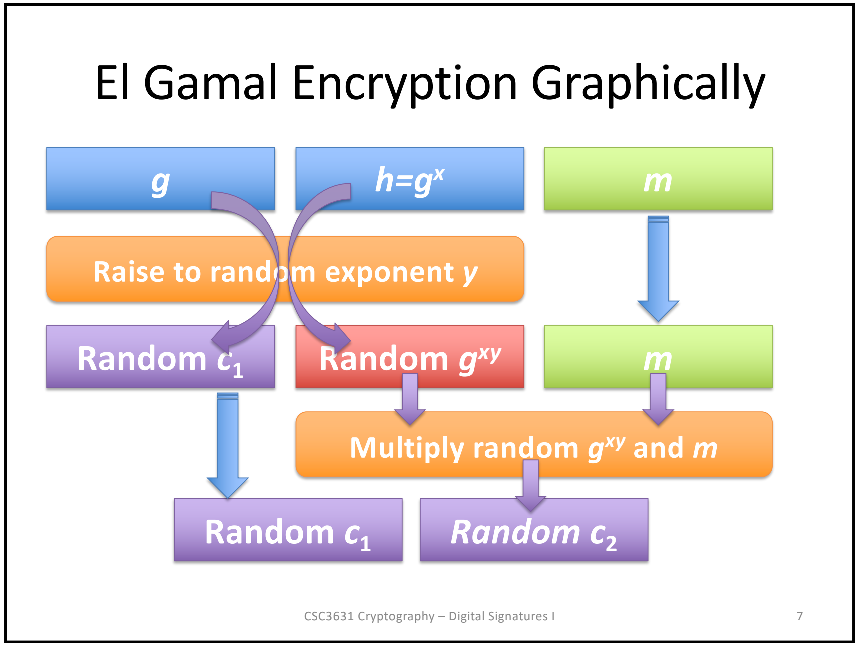
RSA OAEP

Choose random r in $\{0,1\}^n$; Hash-Functions **G** and **H**; XOR

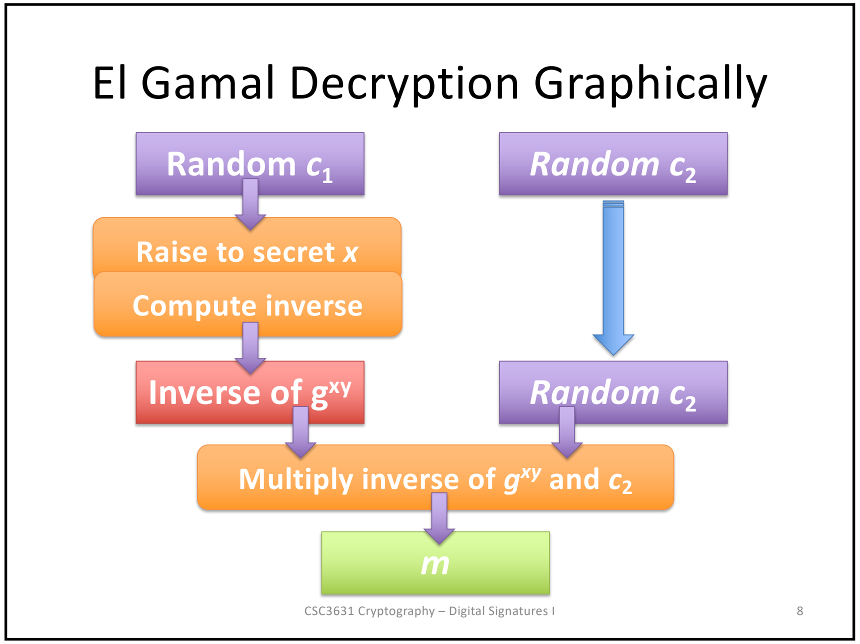



RSA OAEP Decryption
Choose random r in {0,1}ⁿ; Hash-Functions G and H; XOR
m
00000000000…000000000
m’
G
m1
H
r
m1
r xor H(m1)
CSC3631 Cryptography – Asymmetric Encryption II
6

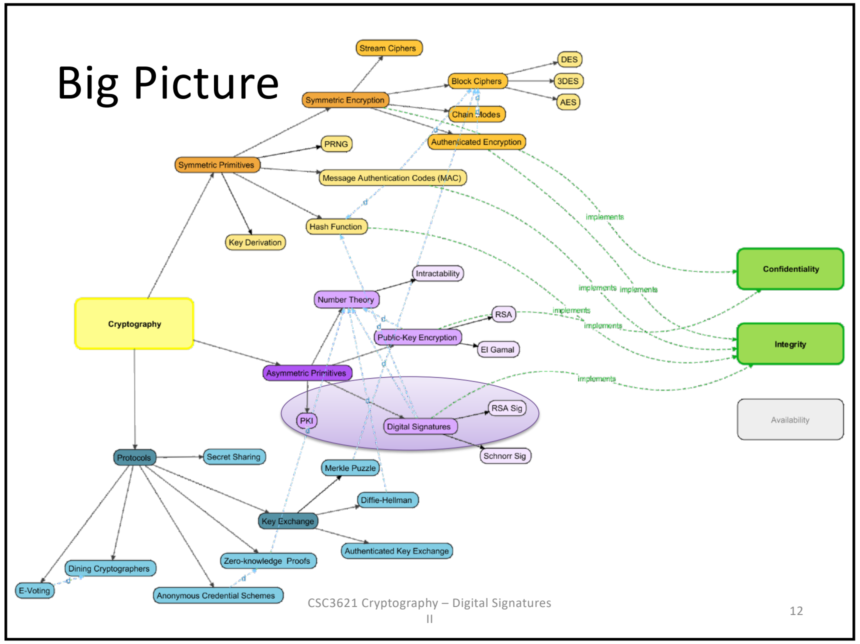
6


El Gamal Encryption Graphically
g
h=g^x
m
Raise to random exponent y
Random c1
Random g^xy
m
Multiply random g^xy and m
Random c1
Random c2
CSC3631 Cryptography – Digital Signatures I
7

7


El Gamal Decryption Graphically
Random c1
Random c2
Raise to secret x
Compute inverse
Inverse of g^xy
Random c2
Multiply inverse of g^xy and c2
m
CSC3631 Cryptography – Digital Signatures I
8

8


Big Picture
Stream Ciphers
DES
Block Ciphers
3DES
Symmetric Encryption
AES
Chain Modes
Authenticated Encryption
PRNG
Symmetric Primitives
Message Authentication Codes (MAC)
implements
Hash Function
Key Derivation
Confidentiality
Intractability
Number Theory
implements
implements
Cryptography
RSA
implements
implements
Public-Key Encryption
Integrity
El Gamal
Asymmetric Primitives
implements
RSA Sig
PKI
Digital Signatures
Availability
Protocols
Secret Sharing
Schnorr Sig
Merkle Puzzle
Diffie-Hellman
Key Exchange
Authenticated Key Exchange
Zero-knowledge Proofs
Dining Cryptographers
E-Voting
Anonymous Credential Schemes
CSC3621 Cryptography – Digital Signatures II
12

12

Roadmap

- Digital Signatures
  - Concepts and characteristics
  - Existential Unforgeability
- RSA Signatures
  - Textbook RSA and its Insecurity
  - Hashed RSA

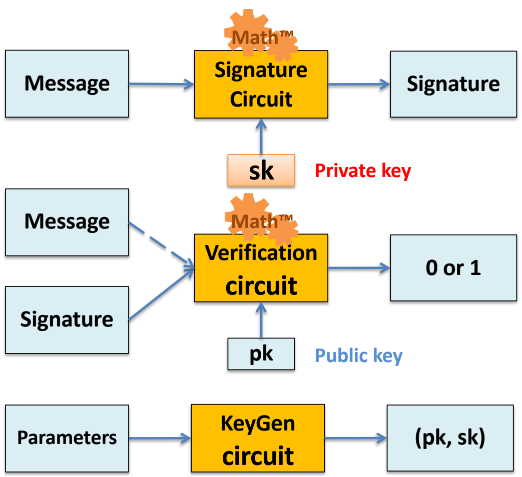
Goal for today:

- **What are digital signatures?**
- **How is the popular RSA Signature Scheme realized?**

Imitating a Hand-written Signature

What properties should a signature scheme have?

Structure



Characteristics

Goal: integrity – Message came from sender & is unmodified

Public verifiability: Everybody with access to pk can verify a signature.

Transferability: One can convince others of the signature's validity.

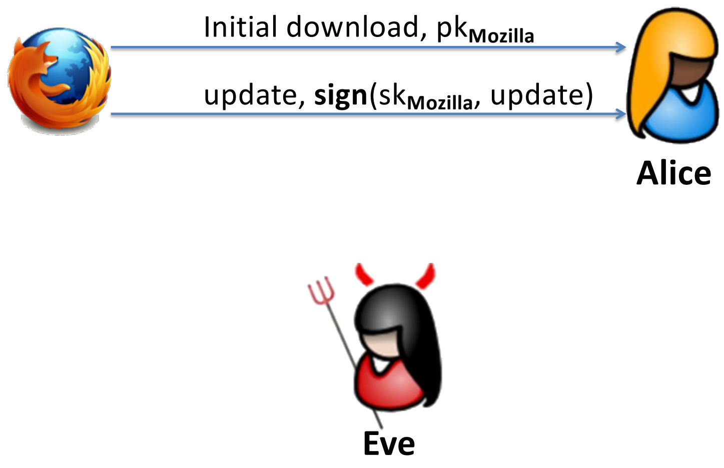
Non-repudiation: Alice cannot repudiate that she has signed the message.

Key authenticity: Publish pk by distributing it with integrity.

Digital Signatures vs. MACs

Signatures	MACs
No pre-shared secret	Need key exchange
Keys independent of sender	Secret key for each pair of parties
Anyone who wants to verify the signature can do so	Only the dedicated partner can verify.
Only a single private key to keep secret	Large number of keys needed
Non-repudiation	Deniable
	2-3 orders of magnitude faster than signature schemes

Example: Update Distribution



[Firefox logo from Mozilla.org, as example only. No trademark implications intended.]

What's the problem here?

Roadmap

- Digital Signatures
  - Concepts and characteristics
  - **Existential Unforgeability**
- RSA Signatures
  - Textbook RSA and its Insecurity
  - Hashed RSA

Goal for today:

- **What are digital signatures?**
- **How is the popular RSA Signature Scheme realized?**

Existential Unforgeability I

No adversary should be able to forge any signature.

Setup: Generate keypair (*pk*, *sk*)

Inputs to Adversary **A:** *pk*, access to $\textbf{Sign}_{sk}()$

A gets signatures on an arbitrary set of messages *m* in *Q*.

Output by **A**: message-signature pair (m^*, σ)

Success criterion for **A:** $\textbf{verify}_{pk}(m^*, \sigma) = \textbf{1}$

m^* not in *Q*.

Existential Unforgeability II

A signature scheme is **existentially unforgeable under an adaptive chosen-message attack** if all probabilistic and polynomial-time adversaries A only have negligible success probability.

Summary

Goal: Integrity

With **public verifiability**, **transferability** and **non-repudiation**.

Remember: Key distribution must be **authentic**!

Key security property: Existential Unforgeability

Roadmap

- Digital Signatures
  - Concepts and characteristics
  - Existential Unforgeability
- **RSA Signatures**
  - Textbook RSA and its Insecurity
  - Hashed RSA

Goal for today:

- **What are digital signatures?**
- **How is the popular RSA Signature Scheme realized?**

Building RSA Signatures

Number Theory

- $(Z_N)^*$ — $N = pq$
- Inverse — e^{-1} (mod N)
- e^{th} root $c^{1/e}$ — $x^e = c$
- $\varphi(N) := |(Z_N)^*|$
- $x^{\varphi(N)}=1$ (**mod** N)

Intractable (Hard) Problems

- Factoring — Given large N: find factors p and q
- RSA Problem (~e^{th} root in $(Z_N)^*$) — Given N, e, y: find x s.t. $x^e = y$ (**mod** N)

Textbook RSA Signature (Pseudo-random Trapdoor Permutation)

- RSAKeyGen — $N=pq$, $d=e^{-1}$ **mod** $\varphi(N)$
- Signing — $\sigma = m^d$ (**mod** N)
- Verification — $m = \sigma^e$ (**mod** N)

The RSA Assumption

Recall: What's the basis of the RSA crypto system?

Setup: (N, e, d) ← **GenRSA**(1^n), where $e \cdot d = 1$ **mod** $\varphi(N)$

Choose y from $(\mathbf{Z}_N)^*$

Input for Adversary **A**: N, e, y

Output of Adversary **A**: x in $(\mathbf{Z}_N)^*$

Adversary A success: if $x^e = y$ (**mod** N)

The RSA problem is **hard** relative to GenRSA if all probabilistic and polynomial-time adversaries **A** only have negligible success probability.

RSA Key Generation

Recall: How to create a strong setting for RSA?

GenRSA(1^n)

Input: key length n

Generate two large n-bit **distinct primes** p and q

Compute $N = p \cdot q$ and $\varphi(N) = (p-1) \cdot (q-1)$

Choose a random integer e, **gcd**$(e, \varphi(N)) = 1$

Compute e's inverse d: $d \cdot e = 1$ (**mod** $\varphi(N)$)

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

Textbook RSA Signatures

KeyGen: $pk=(N, e)$, $sk=(N, d)$ ← **GenRSA**(1^n)

Sign: Given $sk=(N, d)$ and message m:

$$\sigma = m^d \ (\textbf{mod } N)$$

Verify: Given $pk=(e, N)$ and signature σ:

$$m = \sigma^e \ (\textbf{mod } N)$$

How Secure are Textbook RSA Signatures?

- Textbook RSA signatures are existentially unforgeable against adaptive chosen message attacks.
- Textbook RSA signatures are existentially unforgeable against passive against key-only attacks.
- Textbook RSA signatures are secure against selective forgeries, yet not existentially unforgeable.
- Textbook RSA signatures are not secure at all, even if the RSA assumption holds.

Roadmap

- Digital Signatures
  - Concepts and characteristics
  - Existential Unforgeability
- RSA Signatures
  - **Textbook RSA and its Insecurity**
  - Hashed RSA

Goal for today:
- **What are digital signatures?**
- **How is the popular RSA Signature Scheme realized?**

No-message Attack

Adversary **A** only has access to $pk=(N, e)$.

How can he mount an attack?

Selected-Message Attack I

Adversary **A** has access to pk=(N, e) and can obtain two signatures from the signer.

How can **A** forge a signature on any chosen message m?

Selected-Message Attack II

Claim: $\sigma = \sigma_1 \cdot \sigma_2$ (**mod** N) is a valid signature on m
Given: $m_2 = m / m_1$

$$
\begin{aligned}
\sigma &= \sigma_1 \cdot \sigma_2 \\
&= m_1^{d} \cdot m_2^{d} && \text{| Def. of RSA sign} \\
&= m_1^{d} \cdot (m/m_1)^{d} && \text{| Structure of } m_2 \\
&= m_1^{d} \cdot (m^{d}/m_1^{d}) && \text{| Exp. rules} \\
&= m_1^{d} \cdot m^{d}\, m_1^{-d} = m^{d}
\end{aligned}
$$

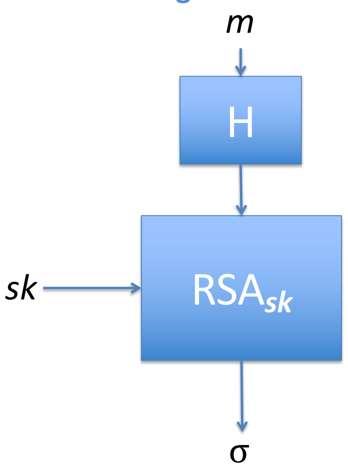
Roadmap

- Digital Signatures
  - Concepts and characteristics
  - Existential Unforgeability
- RSA Signatures
  - Textbook RSA and its Insecurity
  - **Hashed RSA**

Goal for today:
- **What are digital signatures?**
- **How is the popular RSA Signature Scheme realized?**

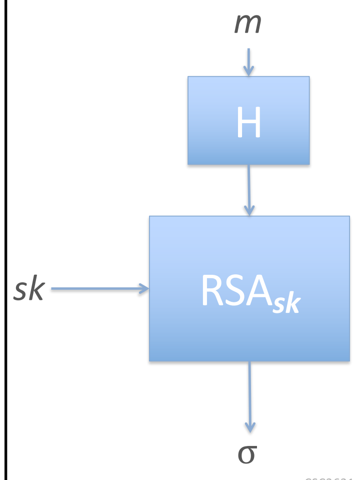
Hashed RSA

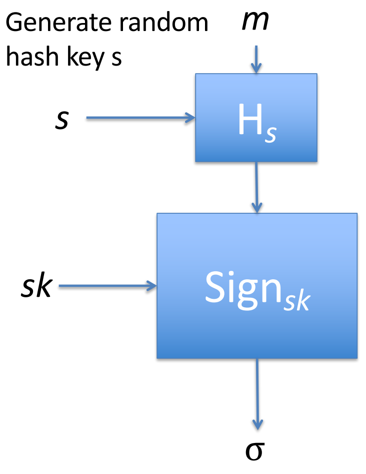
How to get an actual signature scheme out of RSA?



Hashed RSA can be proven to be **existentially unforgeable under an adaptive chosen-message attack** under the RSA assumption and in the **random oracle model**.

Hash-and-Sign Paradigm





Redundancy Methods

PKCS #1

- Hashing
- Strict Integer-to-octet-string encoding

ISO/IEC 9796

- Padding
- Extension
- Redundancy Function

Summary

RSA can form a signature scheme.

Sign: Given $sk=(N, d)$ and m: $\sigma = m^d \pmod{N}$

Verify: Given $pk=(e, N)$ and σ: $m = \sigma^e \pmod{N}$

Textbook RSA is **completely insecure**.

Hash-and-sign is the way forward.