Post-Quantum Cryptography #2

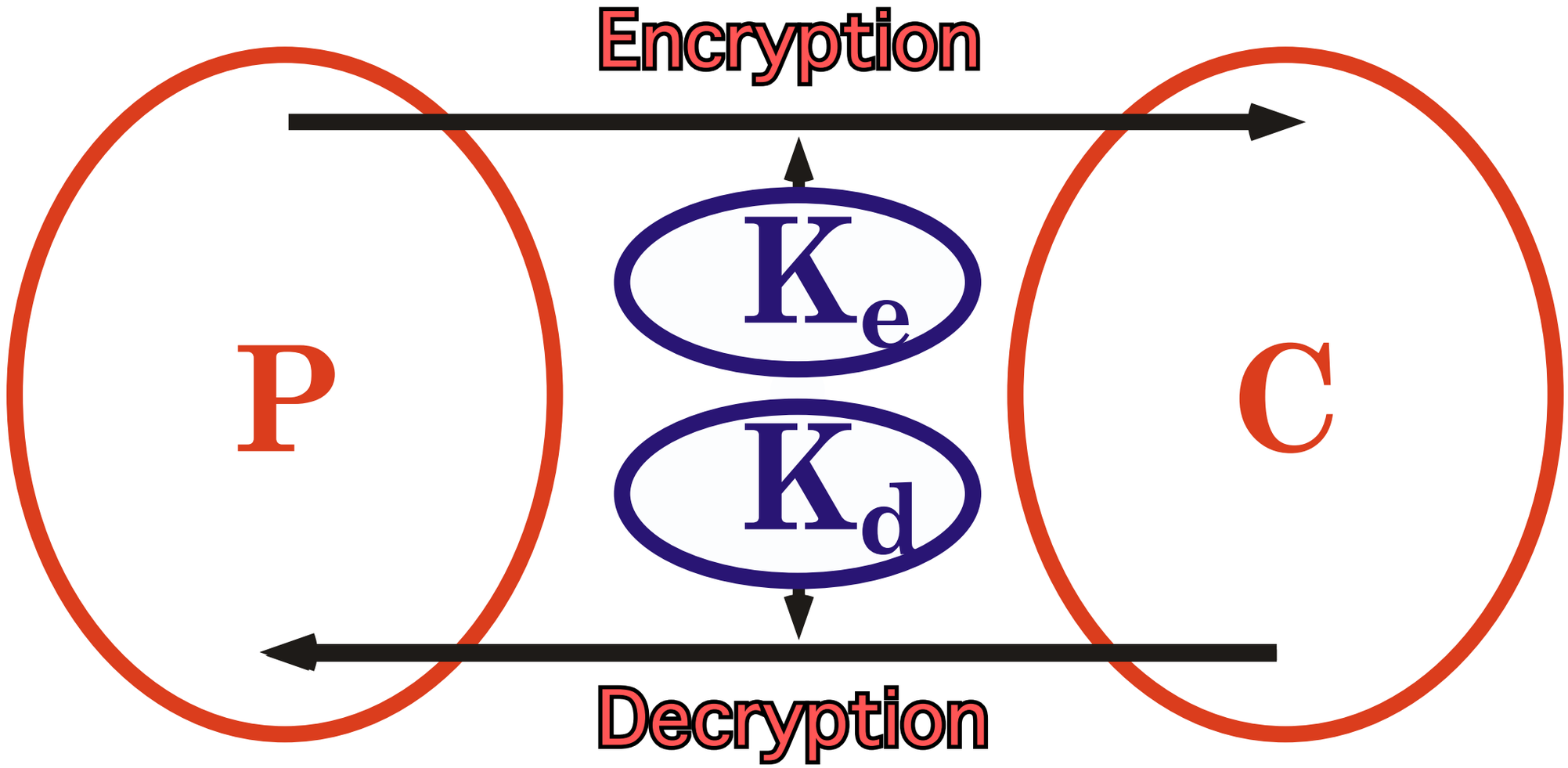
Prof. Claude Crépeau
McGill University

Post-Quantum Cryptography

- Finite Fields based cryptography
  - Codes
  - Multi-variate Polynomials
- Integers based cryptography
  - Approximate Integer GCD
  - Lattices

Public Key Encryption

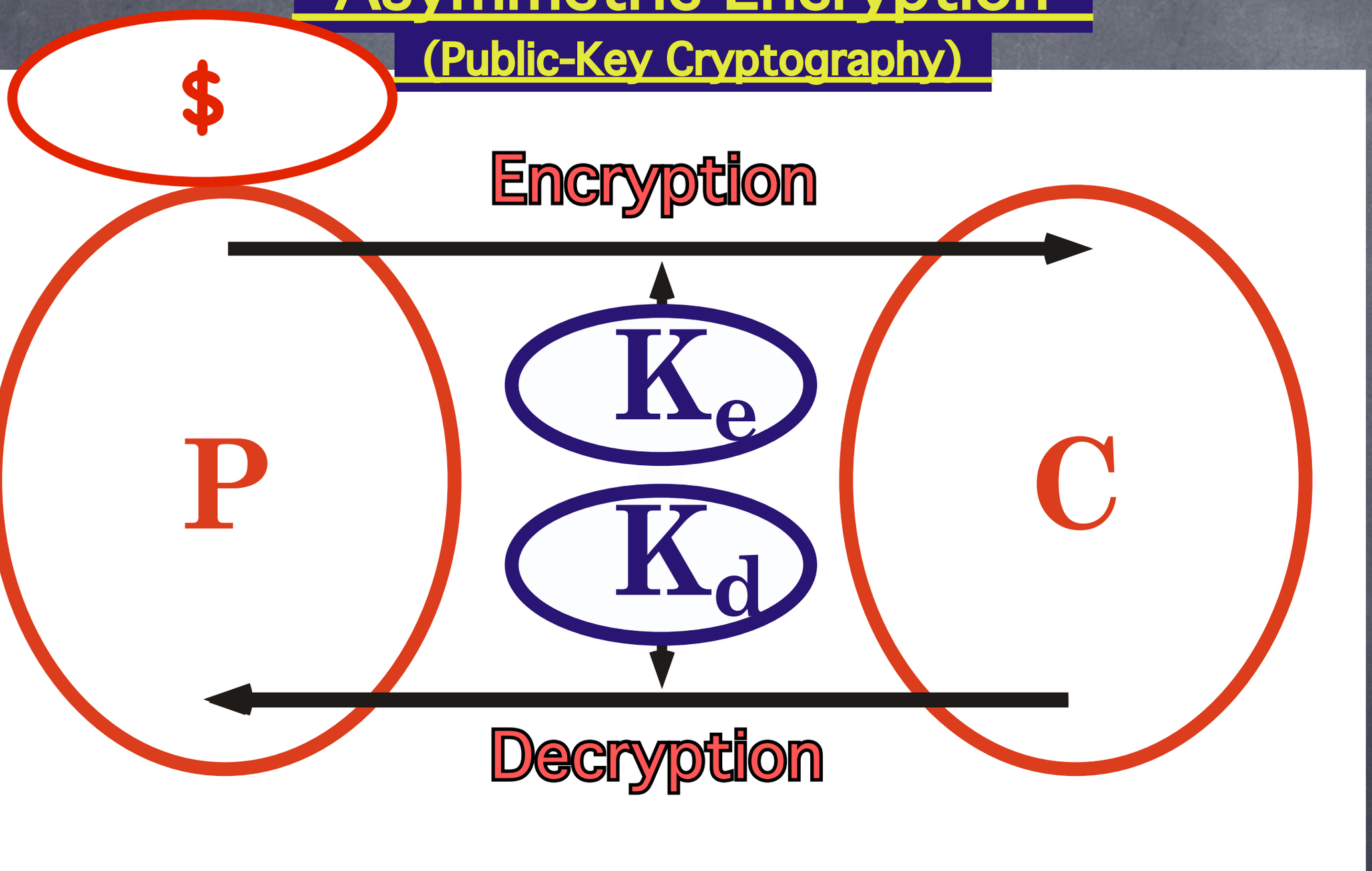
Asymmetric Encryption
(Public-Key Cryptography)



Complexity Theoretical Security

Asymmetric Encryption

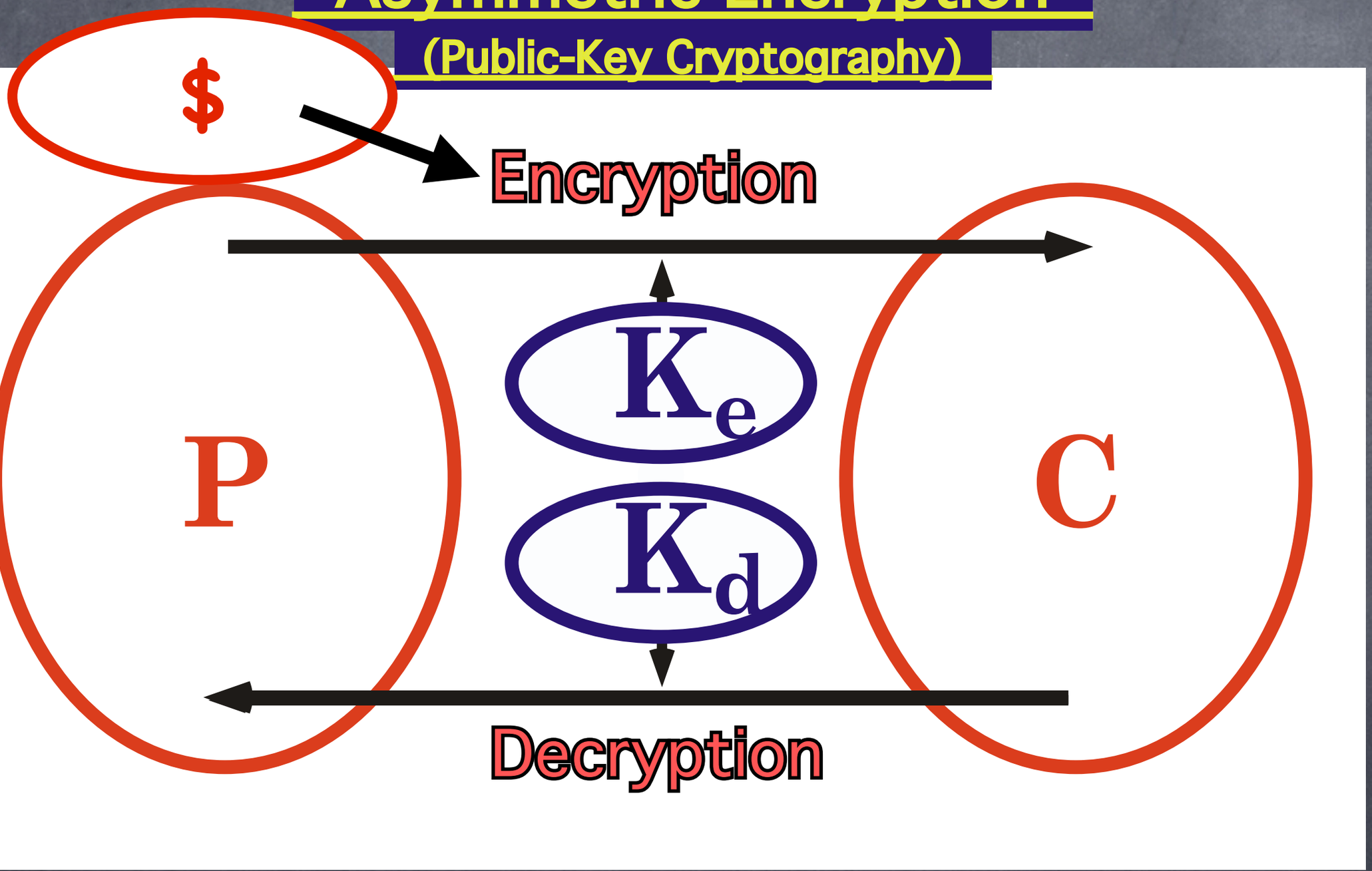
(Public-Key Cryptography)



Complexity Theoretical Security

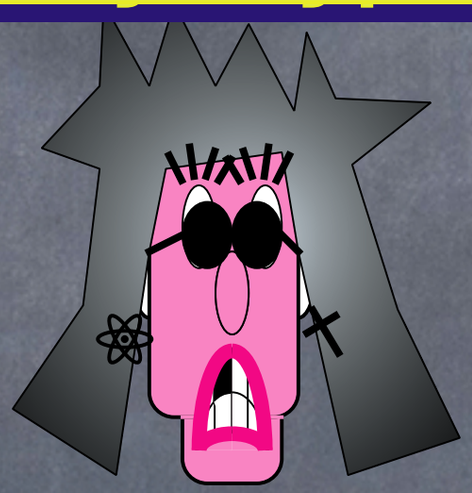
Asymmetric Encryption

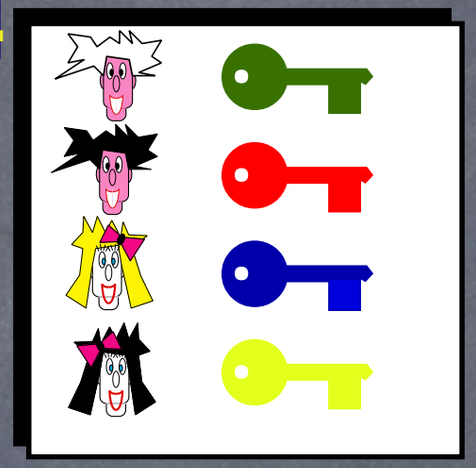
(Public-Key Cryptography)

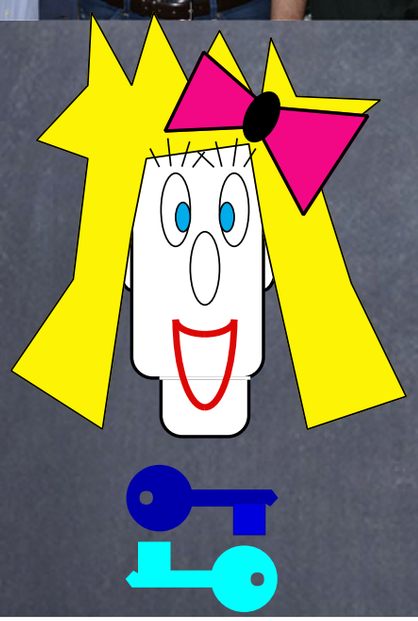


Complexity Theoretical Security

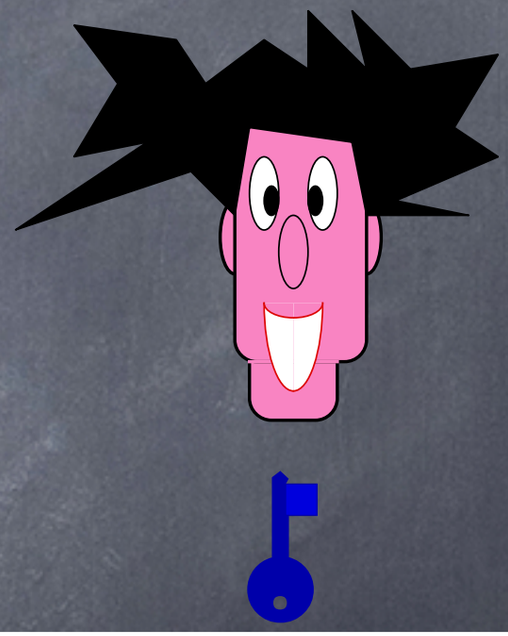
Public-Key Cryptography



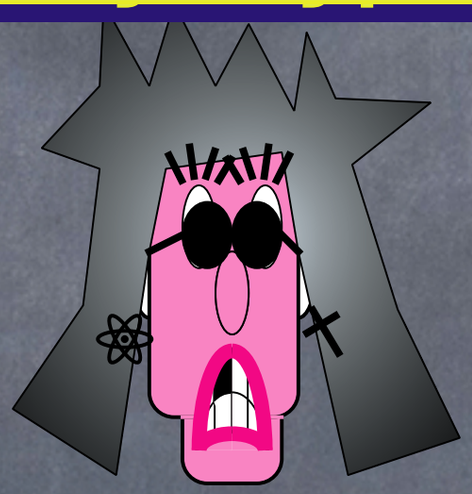


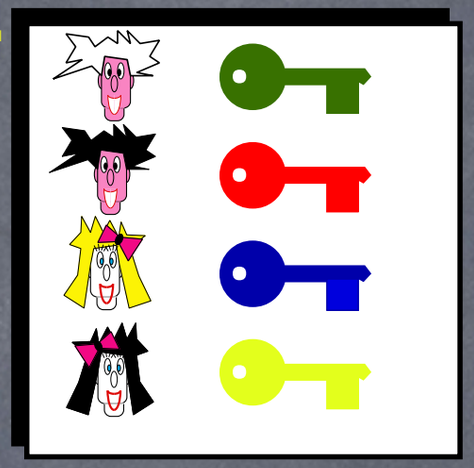


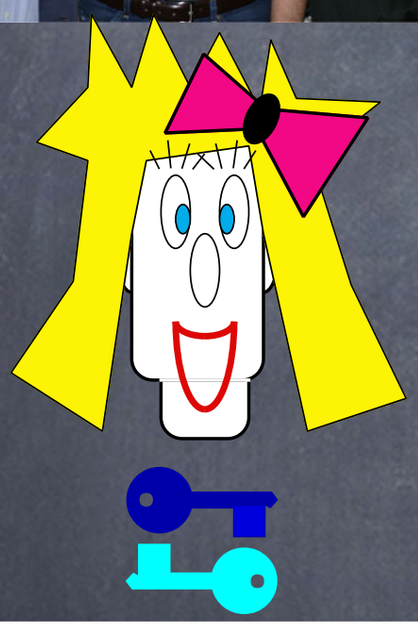
8RdewtU5qkLa$es!T9@



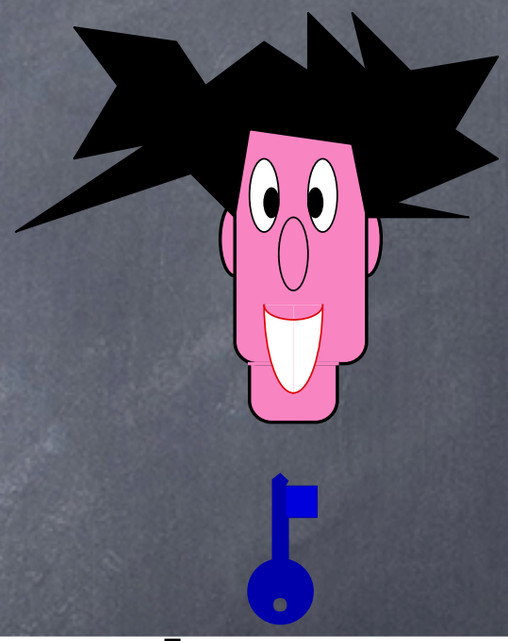
Public-Key Cryptography



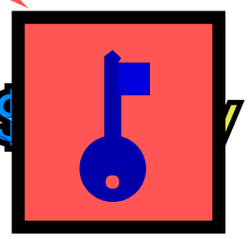




8RdewtU5qkLa$es!T9@

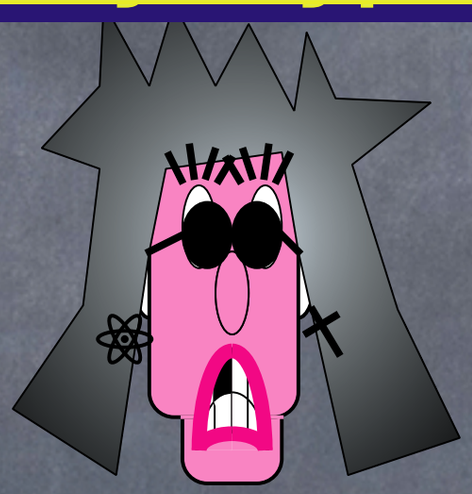


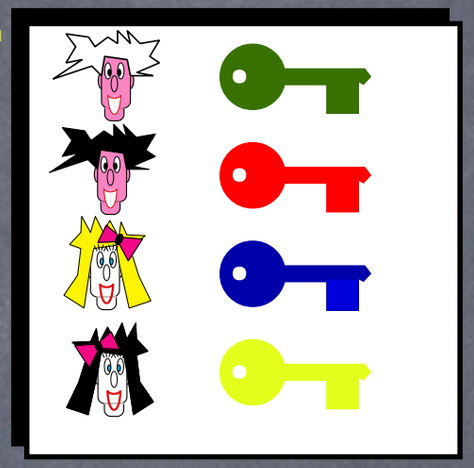
Encryption

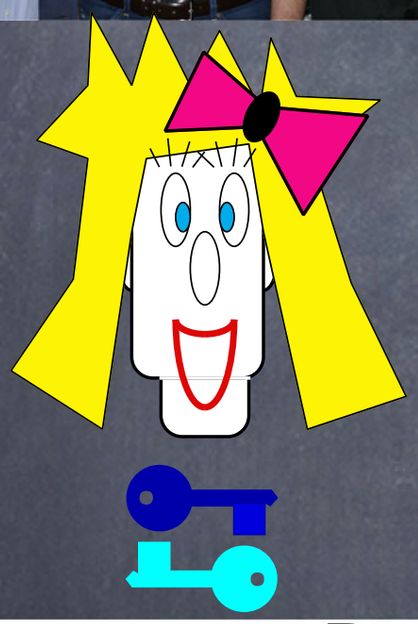


8RdewtU5qkLa$ [illegible] me ?

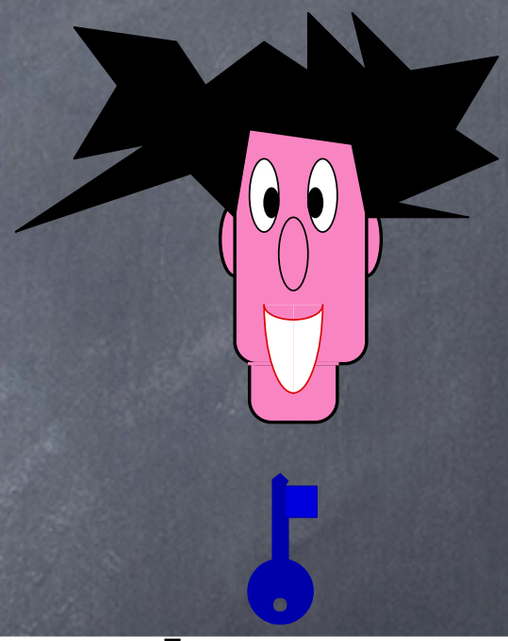
Public-Key Cryptography







8RdewtU5qkLa$es!T9@



Decryption

Will you marry m es!T9@

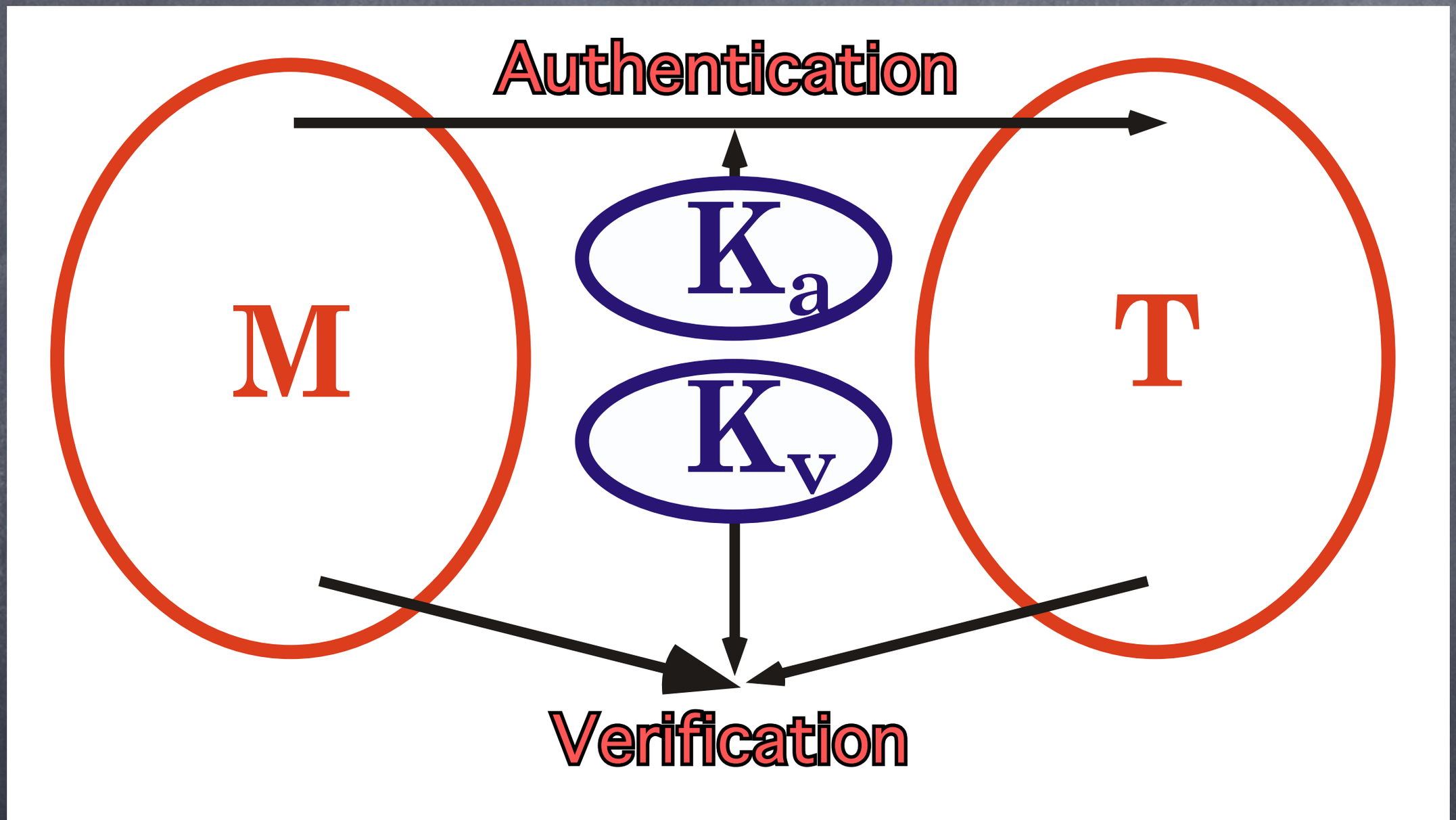
Encryption

8RdewtU5qkLa$ y me ?

Digital Signatures

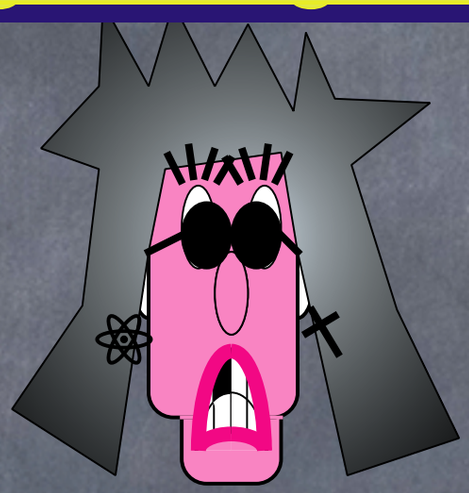
Asymmetric Authentication

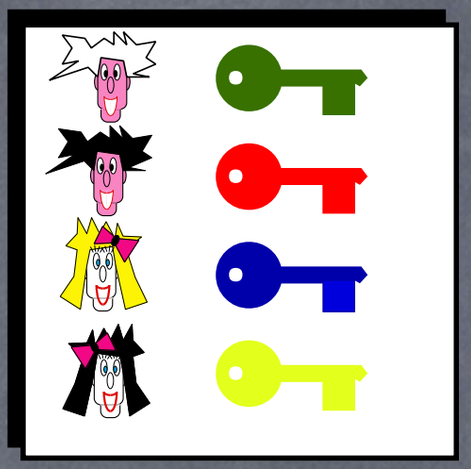
(Digital Signature Scheme)

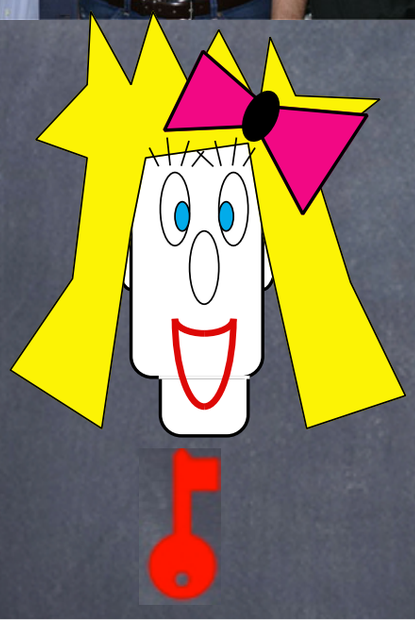


Complexity Theoretical Security

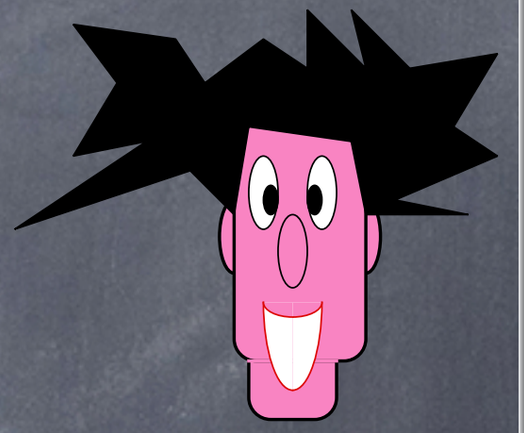
Digital Signature



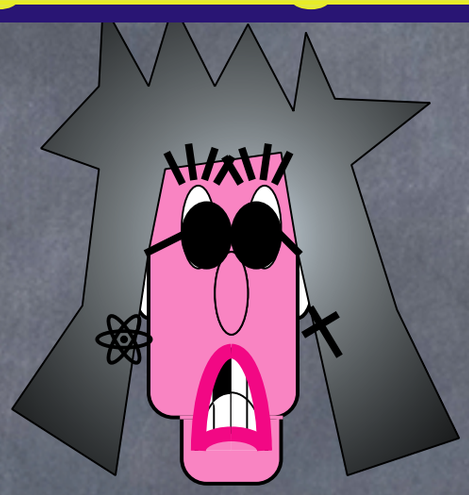


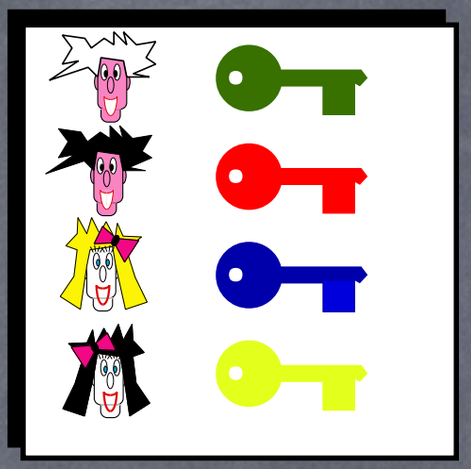


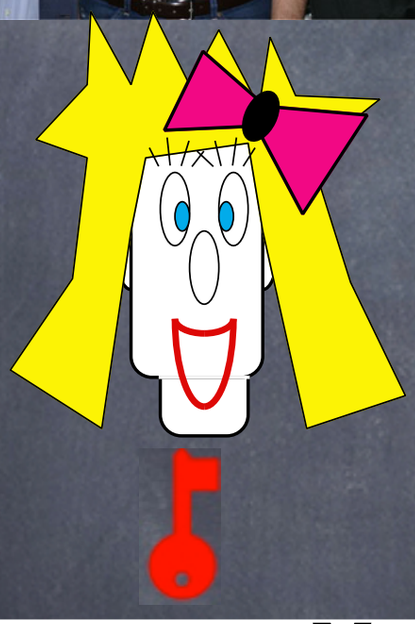
8RdewtU5qkLa$es!T9@
Will you marry me ?



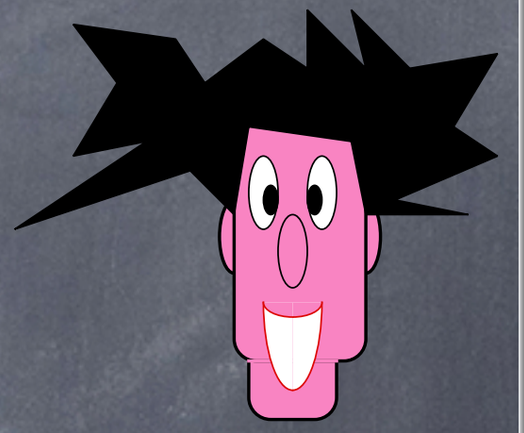
Digital Signature

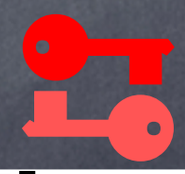






8RdewtU5qkLa$es!T9@
Will you marry me ?

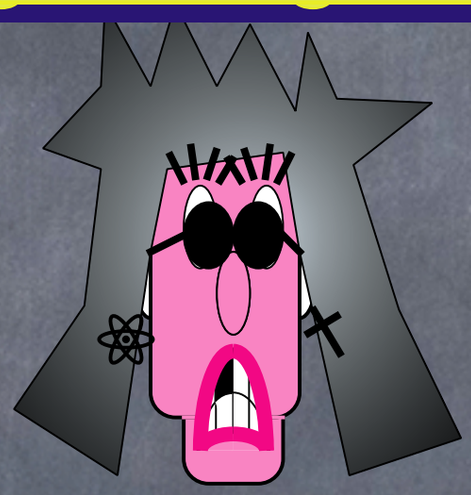


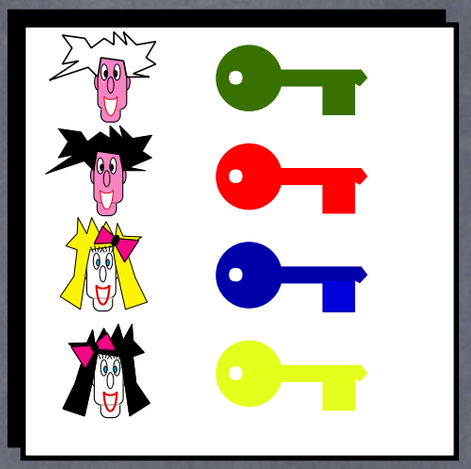


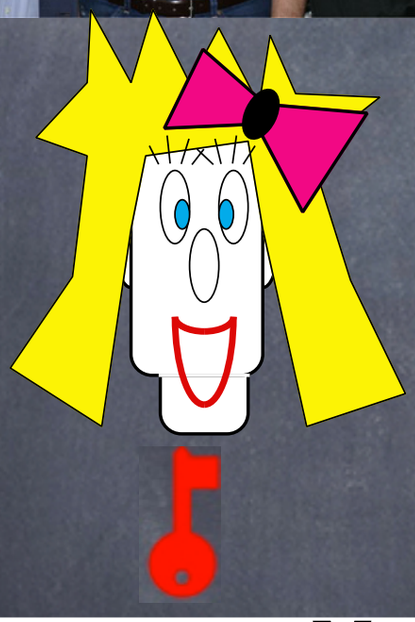
Authentication

8RdewtU5qkLa$ me ?

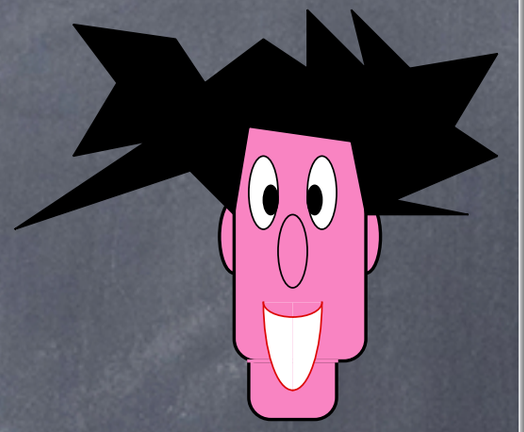
Digital Signature







8RdewtU5qkLa$es!T9@
Will you marry me ?



Verification

VALID es!T9@
ry me ?

Authentication

8RdewtU5qkLa$ y me ?

Code Equivalence

Code Equivalence

- Two [n,k,d] linear codes C,C' are (permutation) equivalent if there exists a kxk non-singular matrix S & an nxn permutation matrix P s.t.

Code Equivalence

- Two [n,k,d] linear codes C,C' are (permutation) equivalent if there exists a kxk non-singular matrix S & an nxn permutation matrix P s.t.

$$G' = SGP$$

Code Equivalence

- Two [n,k,d] linear codes C,C' are (permutation) equivalent if there exists a kxk non-singular matrix S & an nxn permutation matrix P s.t.

$$G' = SGP$$

- the codewords of C and C' have exactly all the same weights

Code Equivalence

Code Equivalence

- Let C' be an [n,k,d] linear code equivalent to a code C.

Code Equivalence

- Let C' be an [n,k,d] linear code equivalent to a code C.
- Let Cor:$\{0,1\}^n \rightarrow C$ be an efficient nearest codeword error-correcting procedure for C (upto $d-1/2$ errors)

Code Equivalence

- Let C' be an [n,k,d] linear code equivalent to a code C.
- Let $Cor:\{0,1\}^n \to C$ be an efficient nearest codeword error-correcting procedure for C (upto $d-1/2$ errors)
- Define $C'or(w):=Cor(wP^{-1})P$,

Code Equivalence

- Let C′ be an [n,k,d] linear code equivalent to a code C.
- Let $Cor:\{0,1\}^n \to C$ be an efficient nearest codeword error-correcting procedure for C (upto $\frac{d-1}{2}$ errors)
- Define $C'or(w) := Cor(wP^{-1})P$,
- then $C'or:\{0,1\}^n \to C'$ is an efficient nearest codeword error-correcting procedure for C′ (upto $\frac{d-1}{2}$ errors)

McEliece Cryptosystem

McEliece Cryptosystem

- Let $G \in_R Goppa_t$, $S \in_R \mathbb{F}_2^{k \times k}$, & $P \in_R Perm$ be the private-key, & $G' = SGP$ be the public-key,

McEliece Cryptosystem

- Let $G \in_r Goppa_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,

- Let $e \in_r \{\text{error vector of weight } t\}$ & $m \in \{0,1\}^k$ a plaintext Let $w = mG' + e$ be a ciphertext.

McEliece Cryptosystem

- Let $G \in_r Goppa_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,

- Let $e \in_r \{\text{error vector of weight } t\}$ & $m \in \{0,1\}^k$ a plaintext let $w = mG' + e$ be a ciphertext.

- Given (only) G', w finding

McEliece Cryptosystem

- Let $G \in_r Goppa_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,

- Let $e \in_r \{\text{error vector of weight } t\}$ & $m \in \{0,1\}^k$ a plaintext let $w = mG' + e$ be a ciphertext.

- Given (only) G', w finding

$$c' = C'or(w) \text{ is difficult.}$$

Niederreiter Cryptosystem

Niederreiter Cryptosystem

- Let $G \in_R GRS_t$, $S \in_R \mathbb{F}_2^{k \times k}$, & $P \in_R Perm$ be the private-key, & $G' = SGP$ be the public-key,

Niederreiter Cryptosystem

- Let $G \in_r GRS_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,
- Let $m \in \{\text{error vector of weight } t\}$ a plaintext & $c' \in_r C'$ Let $w = c' + m$ be a ciphertext.

Niederreiter Cryptosystem

- Let $G \in_r GRS_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,

- Let $m \in \{$error vector of weight $t\}$ a plaintext & $c' \in_r C'$ let $w = c' + m$ be a ciphertext.

- Given (only) G', w finding

Niederreiter Cryptosystem

- Let $G \in_r GRS_t$, $S \in_r \mathbb{F}_2^{k \times k}$, & $P \in_r Perm$ be the private-key, & $G' = SGP$ be the public-key,

- Let $m \in \{\text{error vector of weight } t\}$ a plaintext & $c' \in_r C'$ let $w = c' + m$ be a ciphertext.

- Given (only) G', w finding

$$c' = C'or(w) \text{ is difficult.}$$

Both Cryptosystems

Both Cryptosystems

- Let $G \in_R GRS/Goppa_t$, $S \in_R \mathbb{F}_2^{k \times k}$, & $P \in_R Perm$ be the private-key, & $G' = SGP$ be the public-key, $e \in \{$error vector of weight $t\}$ and let $w = c + e$ for $c \in C(G')$.

Both Cryptosystems

- Let $G \in_R GRS/Goppa_t$, $S \in_R \mathbb{F}_2^{k \times k}$, & $P \in_R Perm$ be the private-key, & $G' = SGP$ be the public-key, $e \in \{$error vector of weight $t\}$ and let $w = c + e$ for $c \in C(G')$.

- Given G, S, P, w finding $c = Cor(w)$ and $e = w - c$ is easy.

	(n,t)	(2048,32)	(2048,40)	(4096,22)	(4096,45)
McEliece scheme	plaintext size[1]	1928	1888	4024	3904
	ciphertext size[1]	2048	2048	4096	4096
	encryption rate[2]	176	222	145	192
	decryption rate[2]	1780	2260	600	1650
Niederreiter scheme	plaintext size[1]	232	280	192	352
	ciphertext size[1]	352	440	264	540
	encryption rate[2]	360	370	320	340
	decryption rate[2]	13600	16700	9800	16900
	public key size[3]	73 KB	86 KB	123 KB	234 KB
	key generation[3]	$6.70\,10^7$	$9.55\,10^7$	$7.93\,10^7$	$23.1\,10^7$
	security bits[3][4]	91	98	93	140

[1] plaintext and ciphertext sizes in bits
[2] in cycles per plaintext bytes, Intel Core 2
[3] common to both schemes (number of cycles on a processor Intel Core 2)
[4] $\log_2$ of the non-quantum binary workfactor

Table 4. McEliece and Niederreiter encryption scheme

Families of Codes

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

- (Generalized) Reed-Solomon codes,

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

- (Generalized) Reed-Solomon codes,
- concatenated codes,

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

- (Generalized) Reed-Solomon codes,
- concatenated codes,
- elliptic codes,

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

- (Generalized) Reed-Solomon codes,
- concatenated codes,
- elliptic codes,
- Reed-Muller codes,

Nicolas Sendrier

Families of Codes

Binary Goppa codes seem safe, but not

- (Generalized) Reed-Solomon codes,
- concatenated codes,
- elliptic codes,
- Reed-Muller codes,
- Convolutional codes

Nicolas Sendrier

Code based cryptography

Code based cryptography

- Courtois, Finiasz and Sendrier signature scheme

Code based cryptography

- Courtois, Finiasz and Sendrier signature scheme
- Stern's identification scheme

Code based cryptography

- Courtois, Finiasz and Sendrier signature scheme
- Stern's identification scheme
- Code based PRNG

Code based cryptography

- Courtois, Finiasz and Sendrier signature scheme
- Stern's identification scheme
- Code based PRNG
- Code based hash function

Code based cryptography

§

Post-Quantum Cryptography

- Finite Fields based cryptography
  - ~~Codes~~
  - Multi-variate Polynomials
- Integers based cryptography
  - Approximate Integer GCD
  - Lattices

Multi-variate Poly based cryptography

§

Multi-variate Poly based cryptography

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}^n$.

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_i Q_{ik}x_i^2 + \sum_{i<j} R_{ijk}x_ix_j$

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_i Q_{ik}x_i^2 + \sum_{i<j} R_{ijk}x_ix_j$
- When we are working over $\mathbb{F}=\mathbb{F}_2$, note that $x^2 = x$, so it suffices to consider multilinear polynomials:
  $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_{i<j} R_{ijk}x_ix_j$

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_i Q_{ik}x_i^2 + \sum_{i<j} R_{ijk}x_ix_j$
- When we are working over $\mathbb{F}=\mathbb{F}_2$, note that $x^2 = x$, so it suffices to consider multilinear polynomials:
  $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_{i<j} R_{ijk}x_ix_j$
- In general, finding x from $z=P(x)$ is NP-hard.

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_i Q_{ik}x_i^2 + \sum_{i<j} R_{ijk}x_ix_j$
- When we are working over $\mathbb{F}=\mathbb{F}_2$, note that $x^2 = x$, so it suffices to consider multilinear polynomials:
  $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_{i<j} R_{ijk}x_ix_j$
- In general, finding x from $z=P(x)$ is NP-hard.
- We seek more :
  finding x from $z=P(x)$ being hard on average.

Multi-variate Poly based cryptography

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}_2^n$.

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}_2^n$.
- $z_k = p_k(x) := \sum_i p_{ik} x_i + \sum_{i<j} R_{ijk} x_i x_j$

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}_2^n$.
- $z_k = p_k(x) := \sum_i p_{ik} x_i + \sum_{i<j} R_{ijk} x_i x_j$
- Public-key: P

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}_2^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_{i\leq j} R_{ijk}x_ix_j$
- Public-key: P
- $Enc_P(x)=P(x)$

Multi-variate Poly based cryptography

- $P = (p_1(x_1,\dots,x_n),\dots,p_m(x_1,\dots,x_n))$ for $x=(x_1,\dots,x_n)$ over $\mathbb{F}_2^n$.
- $z_k = p_k(x) := \sum_i P_{ik}x_i + \sum_{i\le j} R_{ijk}x_ix_j$
- Public-key: P
- $Enc_P(x)=P(x)$
- $Dec(z)$= find x s.t. $z=P(x)$ (specific to P's design)

Multi-variate Poly based cryptography

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T.

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T.
- So, $P = T \circ Q \circ S$: $\mathbb{F}^n \to \mathbb{F}^m$, or $P(x) := M_T\, Q(\, M_S x + c_S\,) + c_T$

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T.
- So, $P = T \circ Q \circ S$: $\mathbb{F}^n \to \mathbb{F}^m$, or $P(x) := M_T\, Q(M_S x + c_S) + c_T$
- In any given scheme, the central map Q belongs to a certain class of quadratic maps whose inverse can be computed relatively easily.

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T.
- So, $P = T \circ Q \circ S$: $\mathbb{F}^n \to \mathbb{F}^m$, or $P(x) := M_T\, Q(\, M_S x + c_S\,) + c_T$
- In any given scheme, the central map Q belongs to a certain class of quadratic maps whose inverse can be computed relatively easily.
- $$x = M_S^{-1}\, Q^{-1}(\, M_T^{-1}\, P(x) - c'_T\,) - c'_S$$

  where $c'_T := M_T^{-1}\, c_T$ and $c'_S := M_S^{-1}\, c_S$

Multi-variate Poly based cryptography

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T. So, $P=T \circ Q \circ S$: $\mathbb{F}^n \to \mathbb{F}^m$, or $P(x):=M_T\, Q(\, M_S x+c_S\,) + c_T$

Multi-variate Poly based cryptography

- MPKCs almost always hide a private map Q via composition with secret affine maps S, and T. So, $P = T \circ Q \circ S: \mathbb{F}^n \to \mathbb{F}^m$, or $P(x) := M_T\, Q(\, M_S x + c_S\,) + c_T$

- Private-key: (M_T^{-1}, c'_T), (M_S^{-1}, c'_S), Q^{-1}

$$Dec(y) = M_S^{-1}\, Q^{-1}(\, M_T^{-1}\, y - c'_T\,) - c'_S$$

where $c'_T := M_T^{-1}\, c_T$ and $c'_S := M_S^{-1}\, c_S$

Matsumoto-Inai

Matsumoto-Imai

- Example: (a sort of RSA type system)

Matsumoto-Imai

- Example: (a sort of RSA type system)
- Any single univariate f over $\mathbb{F}_{2^n}$ can be represented by n multivariate algebraic functions $y_i = f_i(x_1, x_2, \ldots, x_n)$ over $\mathbb{F}_2$.

Matsumoto-Imai

- Example: (a sort of RSA type system)
- Any single univariate f over $\mathbb{F}_{2^n}$ can be represented by n multivariate algebraic functions $y_i = f_i(x_1, x_2, \ldots, x_n)$ over $\mathbb{F}_2$.
- $Q(x) := x^{2^a+1}$, $a<n$, over $\mathbb{F}_{2^n}$ such that $\gcd(2^a+1, 2^n-1)=1$ (squaring over $\mathbb{F}_{2^n}$ is actually a linear transform over $\mathbb{F}_2^n$)*

Matsumoto-Imai

- Example: (a sort of RSA type system)
- Any single univariate f over $\mathbb{F}_{2^n}$ can be represented by n multivariate algebraic functions $y_i = f_i(x_1, x_2, ..., x_n)$ over $\mathbb{F}_2$.
- $Q(x) := x^{2^a+1}$, a<n, over $\mathbb{F}_{2^n}$ such that $\gcd(2^a+1, 2^n-1)=1$ (squaring over $\mathbb{F}_{2^n}$ is actually a linear transform over $\mathbb{F}_2^n$)*
- Then there exists $h := (2^a+1)^{-1} \bmod 2^n-1$ such that $Q^{-1}(y)=y^h$ over $\mathbb{F}_{2^n}$

Squaring over $\mathbb{F}_{2^n}$ is Linear over $\mathbb{F}_2$

- $(x_{n-1},\ldots,x_1,x_0)^2$

Squaring over $\mathbb{F}_{2^n}$ is Linear over $\mathbb{F}_2$

- $(x_{n-1},\dots,x_1,x_0)^2$
  $=(x_{n-1}x^{n-1}+\dots+x_1x+x_0)^2 \bmod P(x)$

Squaring over $\mathbb{F}_{2^n}$ is Linear over $\mathbb{F}_2$

- $(x_{n-1},\dots,x_1,x_0)^2$
  $=(x_{n-1}x^{n-1}+\dots+x_1x+x_0)^2 \bmod P(x)$
  $= x_{n-1}x^{2n-2}+\dots+x_1x^2+x_0 \bmod P(x)$

Squaring over $\mathbb{F}_{2^n}$ is Linear over $\mathbb{F}_2$

- $(x_{n-1},\dots,x_1,x_0)^2$
  $=(x_{n-1}x^{n-1}+\dots+x_1x+x_0)^2 \bmod P(x)$
  $= x_{n-1}x^{2n-2}+\dots+x_1x^2+x_0 \bmod P(x)$

$$= \begin{pmatrix} 1 & x^2 & \dots & x^{2n-2} \\ \mid & \mid & \dots & \mid \\ \mathrm{mod} & \mathrm{mod} & \dots & \mathrm{mod} \\ \mid & \mid & \dots & \mid \\ P & P & \dots & P \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \dots \\ x_{n-1} \end{pmatrix}$$

$$= M_{sq}\, x$$

x^{2^i} over $\mathbb{F}_{2^n}$ is linear over $\mathbb{F}_2$

- $(y_{n-1},\dots,y_1,y_0) = (x_{n-1},\dots,x_1,x_0)^{2^i} = M^i_{sq}\ x$

- is a system of n degree 1 equations

$$y_0 = (M^i_{sq})_0\ x \qquad y_1 = (M^i_{sq})_1\ x$$
$$y_2 = (M^i_{sq})_2\ x \quad \dots \quad y_{n-1} = (M^i_{sq})_{n-1}\ x$$

x^{2i+1} over $\mathbb{F}_{2^n}$ is quadratic over $\mathbb{F}_2$

- $(z_{n-1},\dots,z_1,z_0) = (x_{n-1},\dots,x_1,x_0)^{2i+1}$
  $= (y_{n-1},\dots,y_1,y_0)*(x_{n-1},\dots,x_1,x_0)$

is a system of n degree 2 equations

MI vs RSA

MI vs RSA

- Unlike the RSA scheme, the size q^n-1 of the multiplicative group of $\mathbb{F}_{2^n}$ is known, and thus anyone can compute h from 2^a+1.

MI vs RSA

- Unlike the RSA scheme, the size q^n-1 of the multiplicative group of $\mathbb{F}_{2^n}$ is known, and thus anyone can compute h from 2^a+1.
- MI thus based the security of the scheme on the different principle of mapping obfuscation. (à la McEliece)

SFLASH

SFLASH

- The MI scheme was broken by a very clever attack developed by Patarin in 1995.

SFLASH

- The MI scheme was broken by a very clever attack developed by Patarin in 1995.
- Based on an idea of Shamir from 1993, Patarin et al proposed to avoid their own attack by deleting r out of the n equations from the MI public key, and called the resulting scheme SFLASH.

SFLASH

SFLASH

- If we denote the final truncation Π, the SFLASH public key is:

$$P_\Pi = \Pi \circ T \circ Q \circ S$$

SFLASH

- If we denote the final truncation Π, the SFLASH public key is:

$$P_\Pi = \Pi \circ T \circ Q \circ S$$

- Such truncated keys can be used in signature schemes but not in encryption schemes, since they cannot be inverted uniquely.

SFLASH & NESSIE

SFLASH & NESSIE

- The SFLASH scheme was selected in 2003 by the 'new european schemes for signatures integrity and encryption' Consortium as one of only three recommended public key signature schemes, and as the best known solution for low cost smart cards

SFLASH & NESSIE

- The SFLASH scheme was selected in 2003 by the 'new european schemes for signatures integrity and encryption' Consortium as one of only three recommended public key signature schemes, and as the best known solution for low cost smart cards
- The first version of SFLASH, called $SFLASH^{v1}$, had a subtle bug which was discovered by Gilbert and Minier. It was replaced by two versions ($SFLASH^{v2}$ & $v3$).

SFLASH & NESSIE

- The SFLASH scheme was selected in 2003 by the 'new european schemes for signatures integrity and encryption' Consortium as one of only three recommended public key signature schemes, and as the best known solution for low cost smart cards

- The first version of SFLASH, called SFLASH^{v1}, had a subtle bug which was discovered by Gilbert and Minier. It was replaced by two versions (SFLASH^{v2} & $v3$).

- They differ only in their security parameters:
  for SFLASH^{v2} : $q = 2^7$, $n = 37$, $a = 11$ and $r = 11$
  for SFLASH^{v3} : $q = 2^7$, $n = 67$, $a = 33$ and $r = 11$

SFLASH & NESSIE

- The SFLASH scheme was selected in 2003 by the 'new european schemes for signatures integrity and encryption' Consortium as one of only three recommended public key signature schemes, and as the best known solution for low cost smart cards

- The first version of SFLASH, called $SFLASH^{v1}$, had a subtle bug which was discovered by Gilbert and Minier. It was replaced by two versions ($SFLASH^{v2}$ & $v3$).

- They differ only in their security parameters:
  for $SFLASH^{v2}$: $q = 2^7$, $n = 37$, $a = 11$ and $r = 11$
  for $SFLASH^{v3}$: $q = 2^7$, $n = 67$, $a = 33$ and $r = 11$

- Dubois, Fouque, Shamir, Stern broke $SFLASH^{v2}$ & $v3$ in 2007.

Variations

Var.		Meaning	Slows
Plus	+	extra polynomials in the central map	Slows Signatures
Minus	-	remove central or public polynomials	Slows Encryption
Perturb	i	internal perturbation	Slows All
Project	p	Fix a central variable to be 0	Slows Signatures
Vinegar	v	extra variables that can be set arbitrarily	Slows Encryption
Sparse	s	make single-field central map sparse	*General Speedup*

Table 3. A Summary of Major Modifications in MPKCs, cf. [104]

Variations

Scheme	result	SecrKey	PublKey	KeyGen	SecrMap	PublMap
RSA-1024	1024b	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA-$\mathbb{F}_{2^{163}}$	320b	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
PMI+$(136, 6, 18, 8)$	144b	5.5 kB	165 kB	1.1 sec	1.23 ms	0.18 ms
Rainbow $(2^8, 18, 12, 12)$	336b	24.8 kB	22.5 kB	0.3 sec	0.43 ms	0.40 ms
Rainbow $(2^4, 24, 20, 20)$	256b	91.5 kB	83 kB	1.6 sec	0.93 ms	0.74 ms
QUARTZ	128b	71.0 kB	3.9 kB	3.1 sec	11 sec	0.24 ms

Table 2. Current* Multivariate PKCs Compared on a Pentium III 500

*as of 2008

Multi-variate Poly based cryptography

§

Post-Quantum Cryptography

- Finite Fields based cryptography
  - ~~Codes~~
  - ~~Multi-variate Polynomials~~
- Integers based cryptography
  - Approximate Integer GCD
  - Lattices

Cryptographic Money based on hidden codes *(hidden sub-spaces)*

Hidden (Linear) Code

Hidden (Linear) Code

* a linear [n,k,d] code $C \subset \mathbb{F}^n$ over arbitrary finite field $\mathbb{F}$.

Hidden (Linear) Code

* a linear [n,k,d] code $C \subset \mathbb{F}^n$ over arbitrary finite field $\mathbb{F}$.
* a positive integer degree D,

Hidden (Linear) Code

* a linear [n,k,d] code $C \subset \mathbb{F}^n$ over arbitrary finite field $\mathbb{F}$.
* a positive integer degree D,
* $I_{D,C}$ = { degree-D polynomials that vanish on C }.

Hidden (Linear) Code

* a linear [n,k,d] code $C \subset \mathbb{F}^n$ over arbitrary finite field $\mathbb{F}$.
* a positive integer degree D,
* $I_{D,C}$ = { degree-D polynomials that vanish on C }.
* For simplicity, assume we use $\mathbb{F}=\mathbb{F}_2$.

Hidden Code

Hidden Code

* **Lemma A**
  It is possible to sample a uniformly-random element of $I_{D,C}$ in time $O(n^D)$.

Hidden Code

* **Lemma A**
  It is possible to sample a uniformly-random element of $I_{D,C}$ in time $O(n^D)$.

* **Lemma B**
  Fix $C \subset \mathbb{F}_2^n$ and $\beta > 1$, and choose βn independent uniformly-random samples from $I_{D,C}$.
  With probability $1 - 2^{-\Omega(n)}$, the set of points on which they are all zero is exactly C.

Public Q-Money

Public Q-Money

* $P_1(x)$, $P_2(x)$,...,$P_{\beta n}(x)$ define C=Span(G) **(Public-key)**

Public Q-Money

* $P_1(x)$, $P_2(x)$,...,$P_{\beta n}(x)$ define C=Span(G) **(Public-key)**
* $Q_1(x)$, $Q_2(x)$,...,$Q_{\beta n}(x)$ define $C^{\perp}$=Ker(G) **(Public-key)**

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$
* $|\$\rangle = \Sigma_{c\in C} |c\rangle, \qquad [H]^{\otimes n}|\$\rangle = \Sigma_{c'\in C^{\perp}} |c'\rangle$

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$
* $|\$\rangle = \Sigma_{c\in C} |c\rangle, \qquad [H]^{\otimes n}|\$\rangle = \Sigma_{c'\in C^{\perp}} |c'\rangle$
* checking $|\$\rangle$: using $P_1(x),\ldots,P_{\beta n}(x)$, validate that $|\$\rangle$ is made only of states from C and using $Q_1(x),\ldots,Q_{\beta n}(x)$, validate that $[H]|\$\rangle$ is made only of states from $C^{\perp}$.

Public Q-Money

Public Q-Money

* $P_1(x)$, $P_2(x)$,...,$P_{\beta n}(x)$ define C=Span(G) **(Public-key)**

Public Q-Money

* $P_1(x)$, $P_2(x)$,...,$P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x)$, $Q_2(x)$,...,$Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$
* The special structure of $(C,C^{\perp})$, yields an attack for degree 2 polynomials. So D must be at least 3.

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$
* The special structure of $(C,C^{\perp})$, yields an attack for degree 2 polynomials. So D must be at least 3.
* In Q-Money C or $C^{\perp}$ may be sampled once.

Public Q-Money

* $P_1(x), P_2(x),\ldots,P_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q_1(x), Q_2(x),\ldots,Q_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}, i,j \quad P_i(c)=0$ and $Q_j(c')=0$
* The special structure of $(C,C^{\perp})$, yields an attack for degree 2 polynomials. So D must be at least 3.
* In Q-Money C or $C^{\perp}$ may be sampled once.
* Weakens the security. Degree D=4 with sample is as hard as degree 3 without a sample. So they choose D=4.

Hidden Code

* Let $Z_{D,C,\varepsilon}$ be the distribution which sets

$$Z_{D,C,\varepsilon} = \begin{cases} I_{D,C} \text{ with probability } 1-\varepsilon \\ I_{D,\mathcal{R}} \text{ with probability } \varepsilon \end{cases}$$

where $\mathcal{R}$ is a random code of dimension k.

* **Lemma C**

Fix $C \subset \mathbb{F}_2^n$ and $\varepsilon < 1$, let $\beta = 32/(1-\varepsilon)^2$, and choose βn independent samples from $Z_{D,C,\varepsilon}$. Let $\delta = 1/2 + (1-\varepsilon)/4$. With probability $1 - 2^{-\Omega(n)}$ the set of points on which at least $\delta\beta n$ polynomials are zero is exactly C.

Public Q-Money

Public Q-Money

* $P'_1(x), P'_2(x),\ldots,P'_{\beta n}(x)$ define C=Span(G) **(Public-key)**

Public Q-Money

* $P'_1(x), P'_2(x),\ldots,P'_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q'_1(x), Q'_2(x),\ldots,Q'_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**

Public Q-Money

* $P'_1(x), P'_2(x),\ldots,P'_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q'_1(x), Q'_2(x),\ldots,Q'_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C,\ c'\in C^{\perp}$ $P_i(c)=0$ and $Q_j(c')=0$ with probability $\geq \delta$.

Public Q-Money

* $P'_1(x), P'_2(x),\ldots,P'_{\beta n}(x)$ define $C=Span(G)$ **(Public-key)**
* $Q'_1(x), Q'_2(x),\ldots,Q'_{\beta n}(x)$ define $C^{\perp}=Ker(G)$ **(Public-key)**
* $\forall c\in C, c'\in C^{\perp}$ $P_i(c)=0$ and $Q_j(c')=0$ with probability $\geq \delta$.
* Adding misleading polynomials may only make the assumption harder to break...

Cryptographic Money based on hidden codes *(hidden sub-spaces)*