Robust Finite Field Arithmetic for Fault-Tolerant Public-Key Cryptography

Gunnar Gaubatz and Berk Sunar
Worcester Polytechnic Institute
Massachusetts, U.S.A.

Agenda

- Observations and Motivation
- Need for fault tolerance
- Introduction to homomorphic embeddings
- Error model
- Previous ideas, their flaws, and improvements
- Relation to Coding Theory

Some Observations and Motivation

- Current cryptographic keysizes are “computationally secure”
- But: real and tangible threat from side-channel attacks
- Passive attacks have been sufficiently covered in the literature
- Active attacks may prove to be more difficult to defend against
- Need for robust cryptosystems
- How do you prevent somebody from driving a spike (electrical/mechanical) through your chip?

The Famous CRT Attack on RSA

- Bellcore attacks: Boneh, et al. 1996:
  Introduce arbitrary fault into one of the exponentiations, compute GCD(S-S',N)=p --> Modulus is factored.
- Similar attacks on other signature schemes
- Shamir's countermeasure does not always help [BOS02]
- Raised awareness of necessity to verify signature before returning result.
- Problem: How to tell if something went wrong without verifying the signature?
- Need for lightweight fault-tolerance measures to add robustness to finite field arithmetic

Fault Tolerance requires Error Correction

- Need to introduce redundancy into system
- Use meaningful, i.e. large distance, redundancy to spot errors and/or correct them (i.e. codes!)
- Fundamentally different error model: Malicious adversary vs. binary symmetric channel
- Also: computation instead of transmission
- How do we compute with encoded values? Operations need to be preserved
- (Hint: Cyclic and Arithmetic codes possess an arithmetic structure).

Some more Observations

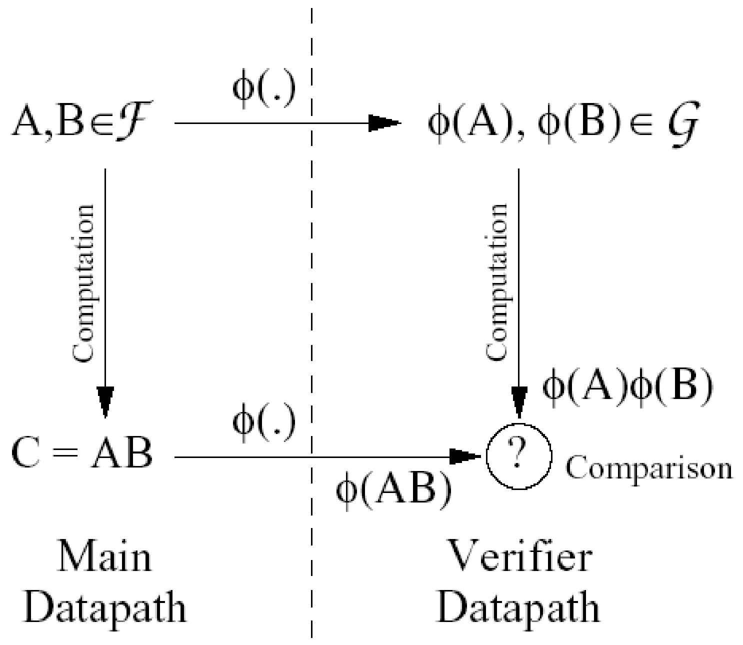
- Underlying arithmetic structure of public key cryptosystems are finite fields, either prime (RSA, DH, ECC) or extension fields (ECC)
- Cyclic and arithmetic codes use similar finite field arithmetic
- Can we make use of finite field structures to achieve fault-tolerance?
- Idea: Homomorphic Embedding using scaling techniques

Homomorphic Embedding

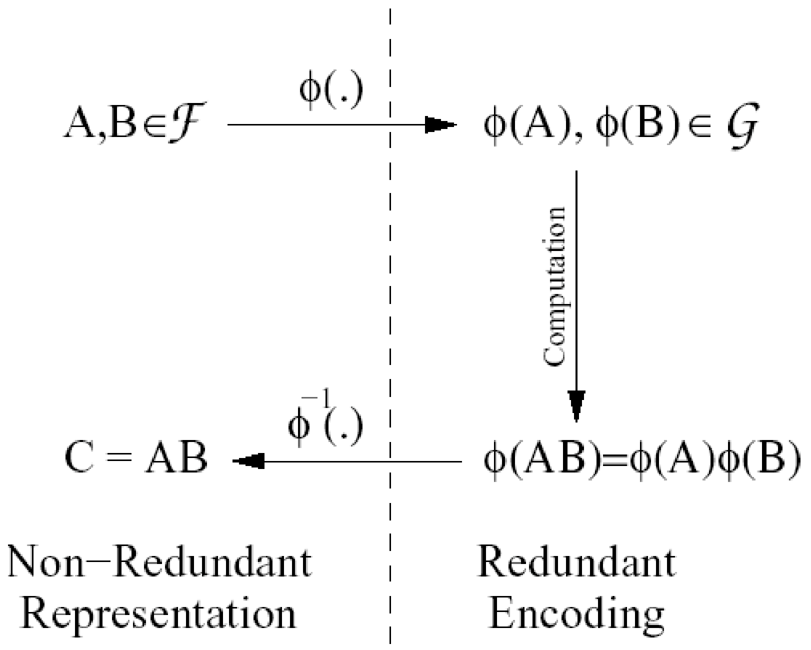
- Use homomorphism to embed field elements into larger (i.e. redundant) ring:
  - Perform all computations in ring w/ redundancy
  - Homomorphism ensures that field operations are preserved in ring as long as there is no fault
  - Faults can be detected after each atomic operation
  - Decode the final result only at the very end if no errors have occurred in any step
- Transformation function $\phi : \mathcal{F} \rightarrow \mathcal{G}$ for elements $a, b \in \mathcal{F}$:
- Preserves additive identity: □(0) = 0
- Preserves addition: □(a) + □(b) = □(a+b)
- Preserves multiplication: □(a·b) = □(a)·□(b)
- Does not necessarily preserve multiplicative inverse

Two strategies

1. $|F| >= |G|$



2. $|F| < |G|$



$A, B \in \mathcal{F}$
$\phi(.)$
$\phi(A), \phi(B) \in \mathcal{G}$
Computation
$C = AB$
$\phi^{-1}(.)$
$\phi(AB)=\phi(A)\phi(B)$
Non−Redundant Representation
Redundant Encoding

Practical Homomorphisms

- Extension fields: □:GF(q^m)->GF($(q^m)^n$)
  - "Natural embedding"
  - Too restrictive to be practical (huge overhead for n>2)
- Ring homomorphisms:
  - Prime field into integer ring:GF(q) -> Zp
  - Extension field into polynomial ring:GF(q^m)->GF(q)[x]
- Homomorphism ensures that ring operations preserve field operations
- Perform all necessary operations in the ring, then transform the result back
- Important: Choose homomorphism with "useful" redundancy, i.e. no naïve embedding

Scaled embedding

- Idea losely based on modulus scaling $m=p\cdot s$
- Scaling of field elements with generator value g: □(a)=g·a □ R
- Goal: Partition the ring into cosets of which only one contains valid symbols
- Error detection by checking for membership
- Scaling factor s determines amount of redundancy $r=\log_2 s$ and form of effective ring modulus
  $m = q\cdot s$
- Choose s such that m has pseudo-Mersenne form $2^n \pm u$ (prime fields) or $x^n \pm u(x)$ (extension fields)

Ring operations

- Addition:
  $\square(a+b) = \square(a) + \square(b) = g(a + b) \bmod m$
- Multiplication:
  $\square(a) \cdot \square(b) = g^2ab \mathrel{!=} \square(a \cdot b) \bmod m$
- Define alternative *-Multiplication:
  $\square(a) * \square(b) = (ga \cdot gb)/g \bmod m = \square(a \cdot b)$
- Note: Division needs to occur strictly before modular reduction step!
- Also: Division is costly and sits on critical path

Error Model

- Assumption of additive errors that may be introduced by an attacker, e.g. through light-attack with focussed laser beam

- Let A= □(a), B= □(b) and e_A,e_B error terms

$$\begin{aligned} C' &= (A + e_A) \star (B + e_B) \\ &= (g^2 \cdot a \cdot b + g(a \cdot e_B + b \cdot e_A) + e_A \cdot e_B)/g \pmod{m} \\ &= C + a \cdot e_B + b \cdot e_A + \frac{e_A \cdot e_B}{g} \pmod{m} \\ &= C + e_C \end{aligned}$$

- Error detection through reduction mod s (but outside of critical path!)

Idempotent Scaled Embedding

- Find idempotent generator $g=g^2 \bmod m$
- Arithmetic AN codes (Proudler 1989)
- Same principle applies to binary fields GF(2^k)->R
- However, one-sided error masking flaw due to distributive law:

$$\begin{aligned} A' &= \phi_i(a) + e_A \\ B &= \phi_i(b) \\ A' \cdot B &= (g \cdot a + e_A)(g \cdot b) \pmod{m} \\ &= g^2 \cdot a \cdot b + g \cdot b \cdot e_A \pmod{m} \\ &= \phi_i((a + e_A)b) \end{aligned}$$

Idempotent AN+B codes

- Exist whenever scaling factor s co-prime with field modulus p
- Define scaling factor g and constant term c:

$$g = (s^{-1} \bmod p)\ s$$
$$c = (p^{-1} \bmod s)\ p$$

- g,c idempotent and $g \cdot c = 0 \bmod m$
- □(a) = ga+c mod m
- Use only for multiplication, since addition is no longer preserved by AN+B codes
- Again, also applies to extension fields

Multiplication

- Re-define *-Multiplication as

$$\begin{aligned} A \star B &= (g \cdot a + c) \cdot (g \cdot b + c) - c \pmod{m} \\ &= g^2 \cdot a \cdot b + c \cdot g(a + b) + c^2 - c \pmod{m} \\ &= \phi_i(a \cdot b) = g(a \cdot b) \pmod{m} \end{aligned}$$

- Conversion from AN to AN+B codes only necessary between heterogeneous operations.
- AN+B codes no longer mask one-sided errors like AN codes do:

$$\begin{aligned} A' \star B &= (g \cdot a + c + e_A)(g \cdot b + c) - c \pmod{m} \\ &= g(a \cdot b) + e_A(g \cdot b + c) \pmod{m} \\ &\equiv e_A \bmod s \end{aligned}$$

Undetectable Errors

- Addition: e_A=-e_B mod s -> e_C = 0 mod s
- Occurs with probability 1/s^2
- Multiplication:

$$
\begin{aligned}
A' \star B' &= (g \cdot a + e_A + c) \cdot (g \cdot b + e_B + c) - c \bmod m \\
&= g(a \cdot b) + e_A(g \cdot b + c) + e_B(g \cdot a + c) + e_A \cdot e_B \bmod m \\
e_{R\star} &= e_A(g \cdot b + c) + e_B(g \cdot a + c) + e_A \cdot e_B \bmod m \ .
\end{aligned}
$$

- Since g=0 mod s and c=1 mod s we have e_R=e_A + e_B + $e_A \cdot e_B$ mod s.
- Undetectable with probability of □(s)/s^2 (□:Euler totient function)

Algorithm Based Fault Tolerance

- Error detection only, no correction. But operation replay a possibility, if operands still available.
- Can be implemented as check for C=0 mod s (possibly beside the main data-path)
- Requires full modular reduction by s (no special prime), but outside the critical path in HW, or only periodically in SW.
- Can only handle transient faults gracefully, permanent faults not correctable

Review: Cyclic Binary and Arithmetic Codes

- Cyclic codes: principal ideals generated by divisors of x^n-1 mod q
- If symbol $(x_0,x_1,\ldots,x_{n-1}) \in C$, then shifted symbol $(x_{n-1},x_0,\ldots,x_{n-2}) \in C$, any valid symbol multiplied by any polynomial is again a valid symbol.
- Arithmetic codes have similar properties, except that carries come into play and require a different interpretation of minimum distance
- Scaled embedding is identical to computing with cyclic or arithmetic codes, if
  - $m=2^n-1$ or $m(x)=x^n-1$ (q=2)
  - s=m/p (p is determined through factorization of m)
- Unfortunately cyclic codes too restrictive for embedding large extension fields of size 100<k<500 (requires large irreducible polynomial)
- Only few suitable parameters (d is design distance):

n	263	359	383	479	503	719	839	863	887	983
k	131	179	191	239	251	359	419	431	443	491
δ	8	9	9	13	9	11	11	9	9	11

Extended Parameter Sets

- Allow m to have pseudo-Mersenne form: $m=2^n \pm u$ or $m(x)=x^n \pm u(x)$, with weight u small, e.g. u=3,5,9,…
- Still very efficient modular reduction, only a few extra adds
- Better chance of finding suitable parameters yielding large irreducible factors or primes (see paper appendix for examples)
- No longer purely cyclic codes, but rather “accumocyclic” codes
- Code family not reported in literature

Conclusions

- Scaled embedding based ring homomorphism are a lightweight method for error detection in finite field arithmetic
- Large possible range of scaling factors for trade-offs between performance and redundancy
- Drawback:
  - Choice of field somewhat dependent on factorization of scaled modulus
  - Not all errors can be detected equally likely
  - Probability of detection is only dependent on error pattern, but not data.
- Future work: Characterization of “Accumocyclic” codes wrt minimum-/ design-distance

Thanks for your attention!

Any questions?