Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis

Bart Mennink (K.U.Leuven)

Joint work with:
Jorge Guajardo (Philips Research Labs)
Berry Schoenmakers (TU Eindhoven)

Financial Cryptography '10, Tenerife, Spain

January 25, 2010

Overview

Preliminaries

Secure Modulo Reduction

Efficiency Analysis

Applications

Conclusions

Threshold Homomorphic Cryptosystems (THCs)

- $[\![x]\!]$ is a *probabilistic encryption*: $[\![x]\!] = \mathsf{Enc}_{pk}(x, r)$ under public key pk and for random r

Threshold Homomorphic Cryptosystems (THCs)

- $[\![x]\!]$ is a *probabilistic encryption*: $[\![x]\!] = \mathrm{Enc}_{pk}(x, r)$ under public key pk and for random r

- Homomorphic properties:
  - Addition: $[\![x]\!][\![y]\!] = [\![x + y]\!]$
  - Multiplication by constant: $[\![x]\!]^c = [\![xc]\!]$
  - Re-randomization: $[\![x]\!][\![0]\!] = [\![x]\!]$

Threshold Homomorphic Cryptosystems (THCs)

- $[\![x]\!]$ is a *probabilistic encryption*: $[\![x]\!] = \mathrm{Enc}_{pk}(x, r)$ under public key pk and for random r

- Homomorphic properties:
  - Addition: $[\![x]\!][\![y]\!] = [\![x + y]\!]$
  - Multiplication by constant: $[\![x]\!]^c = [\![xc]\!]$
  - Re-randomization: $[\![x]\!][\![0]\!] = [\![x]\!]$

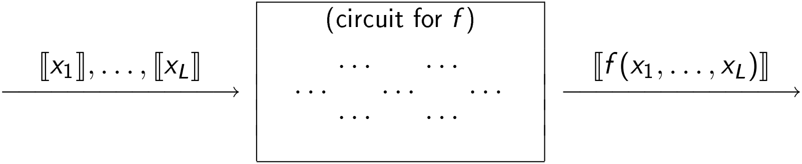
- (t, n)-threshold decryption
  - Private key is shared among n parties such that any t can decrypt

Threshold Homomorphic Cryptosystems (THCs)

- $[\![x]\!]$ is a *probabilistic encryption*: $[\![x]\!] = \text{Enc}_{pk}(x, r)$ under public key pk and for random r
- Homomorphic properties:
  - Addition: $[\![x]\!][\![y]\!] = [\![x + y]\!]$
  - Multiplication by constant: $[\![x]\!]^c = [\![xc]\!]$
  - Re-randomization: $[\![x]\!][\![0]\!] = [\![x]\!]$
- (t, n)-threshold decryption
  - Private key is shared among n parties such that any t can decrypt
- We use the Paillier cryptosystem [Pai99, DJ01]

Secure Function Evaluation based on THCs

- On input of $[\![x_1]\!], \dots, [\![x_L]\!]$ and a function f, the parties jointly compute $[\![f(x_1, \dots, x_L)]\!]$



Secure Function Evaluation based on THCs

- On input of $[\![x_1]\!], \ldots, [\![x_L]\!]$ and a function f, the parties jointly compute $[\![f(x_1, \ldots, x_L)]\!]$

$$\xrightarrow{[\![x_1]\!], \ldots, [\![x_L]\!]} \boxed{\begin{array}{c}\text{(circuit for } f\text{)} \\ \cdots \quad \cdots \\ \cdots \quad \cdots \quad \cdots \\ \cdots \quad \cdots \end{array}} \xrightarrow{[\![f(x_1, \ldots, x_L)]\!]}$$

- Approach based on arithmetic circuits
    - Circuit for f consists of sequential evaluations of $(+, -, *, /)$

Secure Function Evaluation based on THCs (cont.)

- Addition and scalar multiplication by homomorphic properties:
  computation of $[\![x+y]\!]$ and $[\![cx]\!]$ given $[\![x]\!], [\![y]\!], c$
- Multiplication gate [CDN01]: outputs $[\![xy]\!]$ given $[\![x]\!], [\![y]\!]$

Secure Function Evaluation based on THCs (cont.)

- Addition and scalar multiplication by homomorphic properties: computation of $[\![x+y]\!]$ and $[\![cx]\!]$ given $[\![x]\!], [\![y]\!], c$
- Multiplication gate [CDN01]: outputs $[\![xy]\!]$ given $[\![x]\!], [\![y]\!]$
- Our contribution: efficient gate for $[\![x \bmod a]\!]$ given $[\![x]\!], a$
  - Implies a gate for integer division $[\![x \operatorname{div} a]\!]$

Secure Function Evaluation based on THCs (cont.)

- Addition and scalar multiplication by homomorphic properties:
  computation of $[\![x+y]\!]$ and $[\![cx]\!]$ given $[\![x]\!], [\![y]\!], c$
- Multiplication gate [CDN01]: outputs $[\![xy]\!]$ given $[\![x]\!], [\![y]\!]$

- Our contribution: efficient gate for $[\![x \bmod a]\!]$ given $[\![x]\!], a$
  - Implies a gate for integer division $[\![x \operatorname{div} a]\!]$

- Several other efficient gates:
  - Random bit generation gate [CDN01, ST06]:
    outputs $[\![r]\!]$ for random $r \in \{0, 1\}$
  - Comparison gate [DFK+06, GSV07]:
    outputs $[\![x < y]\!]$ given the encrypted bits of x, y
  - Least significant bit gate [ST06]:
    outputs $[\![x \bmod 2]\!]$ given $[\![x]\!]$

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

1. Given $[\![x]\!]$, one decryption of a *blinded version* of x is required

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

1. Given $[\![x]\!]$, one decryption of a *blinded version* of x is required
   - Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$, and $[\![s]\!]$ for random s
   - The blinded encryption $[\![x - r + as]\!]$ is threshold decrypted

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

1. Given $[\![x]\!]$, one decryption of a *blinded version* of x is required
   - Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$, and $[\![s]\!]$ for random s
   - The blinded encryption $[\![x - r + as]\!]$ is threshold decrypted
2. The parties set $\bar{x} = (x - r + as) \bmod a = x - r \bmod a$
   - Notice that $x \equiv \bar{x} + r \bmod a$ and $0 \leq \bar{x} + r < 2a$

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

1. Given $[\![x]\!]$, one decryption of a *blinded version* of x is required
   - Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$, and $[\![s]\!]$ for random s
   - The blinded encryption $[\![x - r + as]\!]$ is threshold decrypted
2. The parties set $\bar{x} = (x - r + as) \bmod a = x - r \bmod a$
   - Notice that $x \equiv \bar{x} + r \bmod a$ and $0 \leq \bar{x} + r < 2a$
3. Correction: the parties compute $[\![c]\!] = [\![a - 1 - \bar{x} < r]\!]$
   - Notice that $c = 0 \iff \bar{x} + r < a$

Secure Modulo Reduction

$$x = (x \operatorname{div} a)a + (x \bmod a), \text{ with } 0 \leq x \bmod a < a$$

- By homomorphic properties, it suffices to determine $x \bmod a$

1. Given $[\![x]\!]$, one decryption of a *blinded version* of x is required
    - Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$, and $[\![s]\!]$ for random s
    - The blinded encryption $[\![x - r + as]\!]$ is threshold decrypted
2. The parties set $\bar{x} = (x - r + as) \bmod a = x - r \bmod a$
    - Notice that $x \equiv \bar{x} + r \bmod a$ and $0 \leq \bar{x} + r < 2a$
3. Correction: the parties compute $[\![c]\!] = [\![a - 1 - \bar{x} < r]\!]$
    - Notice that $c = 0 \iff \bar{x} + r < a$
4. Output $[\![x \bmod a]\!] = [\![\bar{x} + r - ca]\!] = [\![\bar{x}]\!][\![r]\!]/[\![c]\!]^a$

Secure Modulo Reduction (cont.)

- Technical detail: $x - r + as$ should not exceed the Paillier modulus, to prevent wrap-arounds
  - x should be sufficiently small

- Using efficient zero-knowledge proofs, the protocol can be proven secure against *actively malicious* parties (in the security framework of [CDN01])

- How to securely generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$?

How to Securely Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$?

- If $a = 2^{\ell_a}$
  - Write $r = \sum_{i=0}^{\ell_a - 1} r_i 2^i$, with $r_i \in \{0, 1\}$
  - Generate random bits $[\![r_i]\!]$ and output $[\![r]\!] = \prod_{i=0}^{\ell_a - 1} [\![r_i]\!]^{2^i}$

- If $2^{\ell_a - 1} < a < 2^{\ell_a}$

How to Securely Generate $[\![r]\!]$ (bitwise) for $r \in_R [0, a)$?

- If $a = 2^{\ell_a}$
  - Write $r = \sum_{i=0}^{\ell_a - 1} r_i 2^i$, with $r_i \in \{0, 1\}$
  - Generate random bits $[\![r_i]\!]$ and output $[\![r]\!] = \prod_{i=0}^{\ell_a - 1} [\![r_i]\!]^{2^i}$
- If $2^{\ell_a - 1} < a < 2^{\ell_a}$
  - Repeat generating $[\![r]\!]$ for random $r \in [0, 2^{\ell_a})$, until $r < a$
  - At most 2 restarts on average

Efficiency Analysis

	broadcast	round
Ours	$O(nk\ell_a)$	$O(n)$
	$O(n^2 k\ell_a)$	$O(1)$
[DFK$^+$06]	$O(nk\ell_x(\log \ell_x + \ell_a))$	$O(n + \ell_x)$
	$O(nk\ell_x(n + \log \ell_x + \ell_a))$	$O(1)$

n is the number of participants $\quad$ ℓ_x is length of x
k is a security parameter $\quad$ ℓ_a is length of a

- (Broadcast complexity represents the number of bits broadcasted. E.g., for $O(nk\ell_a)$: each party needs to broadcast $O(\ell_a)$ encryptions)
- Always $\ell_a \leq \ell_x$, but often $\ell_a \ll \ell_x$

Efficiency Analysis

	broadcast	round
Ours	$O(nk\ell_a)$	$O(n)$
	$O(n^2k\ell_a)$	$O(1)$
[DFK+06]	$O(nk\ell_x(\log \ell_x + \ell_a))$	$O(n + \ell_x)$
	$O(nk\ell_x(n + \log \ell_x + \ell_a))$	$O(1)$

n is the number of participants $\quad \ell_x$ is length of x
k is a security parameter $\quad \ell_a$ is length of a

- (Broadcast complexity represents the number of bits broadcasted. E.g., for $O(nk\ell_a)$: each party needs to broadcast $O(\ell_a)$ encryptions)
- Always $\ell_a \leq \ell_x$, but often $\ell_a \ll \ell_x$
  - 100 millionaires securely compute their mean fortune
  - $(\llbracket x_1 \rrbracket, \ldots, \llbracket x_{100} \rrbracket) \mapsto \llbracket \frac{x_1 + \cdots + x_{100}}{100} \rrbracket$. Say $x_i < 2^{30}$
  - Here, $x = \sum_{i=1}^{100} x_i$ and $a = 100$, so $\ell_x = 37$ and $\ell_a = 7$

Applications

- Integer division:
  - $x = (x \operatorname{div} a)a + (x \bmod a)$
  - $[\![x \operatorname{div} a]\!] = ([\![x]\!]/[\![x \bmod a]\!])^{1/a}$

Applications

- Integer division:
  - $x = (x \operatorname{div} a)a + (x \bmod a)$
  - $[\![x \operatorname{div} a]\!] = ([\![x]\!]/[\![x \bmod a]\!])^{1/a}$

- Access arbitrary bits of x:
  - $x_i = (x \operatorname{div} 2^i) \bmod 2$

Applications

- Integer division:
  - $x = (x \operatorname{div} a)a + (x \bmod a)$
  - $[\![x \operatorname{div} a]\!] = ([\![x]\!]/[\![x \bmod a]\!])^{1/a}$

- Access arbitrary bits of x:
  - $x_i = (x \operatorname{div} 2^i) \bmod 2$

- Secure computation of statistics:
  - Mean, median, variance, ... require division
  - Concrete example: variance (where $\bar{x} = (x_1 + \cdots + x_L)/L$)

$$\operatorname{var}(x_1, \ldots, x_L) = \frac{1}{L-1} \sum_{i=1}^{L} (x_i - \bar{x})^2$$

Computation of Statistics

$$\mathrm{var}(x_1, \ldots, x_L) = \frac{1}{L-1} \sum_{i=1}^{L} (x_i - \bar{x})^2 = \frac{1}{L(L-1)} \left(\sum_{i=1}^{L} L x_i^2 - \Big(\sum_{i=1}^{L} x_i \Big)^2 \right)$$

- How to compute $[\![\mathrm{var}(x_1, \ldots, x_L)]\!]$ given $[\![x_1]\!], \ldots, [\![x_L]\!]$?

Computation of Statistics

$$\mathrm{var}(x_1, \ldots, x_L) = \frac{1}{L-1}\sum_{i=1}^{L}(x_i - \bar{x})^2 = \frac{1}{L(L-1)}\underbrace{\left(\sum_{i=1}^{L} Lx_i^2 - \Big(\sum_{i=1}^{L} x_i\Big)^2\right)}_{=:V}$$

- How to compute $[\![\mathrm{var}(x_1, \ldots, x_L)]\!]$ given $[\![x_1]\!], \ldots, [\![x_L]\!]$?

  1. Compute $[\![(\sum_{i=1}^{L} x_i)^2]\!]$ and $[\![x_i^2]\!]$ using $L+1$ multiplications
  2. Compute $[\![V]\!] = \left[\!\!\left[\sum_{i=1}^{L} Lx_i^2 - (\sum_{i=1}^{L} x_i)^2\right]\!\!\right]$

Computation of Statistics

$$\mathrm{var}(x_1,\ldots,x_L) = \frac{1}{L-1}\sum_{i=1}^{L}(x_i-\bar{x})^2 = \frac{1}{L(L-1)}\underbrace{\left(\sum_{i=1}^{L} Lx_i^2 - \Big(\sum_{i=1}^{L} x_i\Big)^2\right)}_{=:V}$$

- How to compute $[\![\mathrm{var}(x_1,\ldots,x_L)]\!]$ given $[\![x_1]\!],\ldots,[\![x_L]\!]$?
  1. Compute $[\![(\sum_{i=1}^{L} x_i)^2]\!]$ and $[\![x_i^2]\!]$ using $L+1$ multiplications
  2. Compute $[\![V]\!] = \left[\!\!\left[\sum_{i=1}^{L} Lx_i^2 - (\sum_{i=1}^{L} x_i)^2\right]\!\!\right]$
  3. Compute and output integer division $[\![V \operatorname{div} L(L-1)]\!]$

Computation of Statistics

$$\mathrm{var}(x_1,\ldots,x_L) = \frac{1}{L-1}\sum_{i=1}^{L}(x_i-\bar{x})^2 = \frac{1}{L(L-1)}\underbrace{\left(\sum_{i=1}^{L} Lx_i^2 - \Big(\sum_{i=1}^{L} x_i\Big)^2\right)}_{=:V}$$

- How to compute $[\![\mathrm{var}(x_1,\ldots,x_L)]\!]$ given $[\![x_1]\!],\ldots,[\![x_L]\!]$?

  1. Compute $[\![(\sum_{i=1}^{L} x_i)^2]\!]$ and $[\![x_i^2]\!]$ using $L+1$ multiplications
  2. Compute $[\![V]\!] = \left[\!\!\left[\sum_{i=1}^{L} Lx_i^2 - (\sum_{i=1}^{L} x_i)^2\right]\!\!\right]$
  3. Compute and output integer division $[\![V \operatorname{div} L(L-1)]\!]$

- $O(Lnk)$ broadcast complexity and $O(n)$ rounds

Conclusions

- Modulo reduction: computing $[\![x \bmod a]\!]$ given $[\![x]\!]$ and a
  - Integer division: computation of $[\![x \operatorname{div} a]\!]$
- Applicable to secure computation of statistics (mean, variance, median, range, ...), packing of encrypted data, and many more!
- Our protocols improved performance. We take advantage of the fact that the modulus a is much smaller than x
  - Complexities are independent of the length of x
- Proof of security can be found in the paper (full version)

Questions?

HIDDEN SLIDES!!!

Sub-Protocol: Random Bitwise Value Generation

- Input: a with $2^{\ell_a - 1} < a \leq 2^{\ell_a}$
- For generating $[\![r]\!]$ (bitwise) such that $r \in_R [0, a)$, the n servers do:
  1. Jointly construct ℓ_a random bit encryptions $[\![r_j]\!]$ (note that $r = \sum_{j=0}^{\ell_a - 1} r_j 2^j \in_R [0, 2^{\ell_a}))$
  2. Compute and decrypt $[\![r < a]\!]$. If $r \geq a$, restart protocol
- If $a = 2^{\ell_a}$, no restarts. Otherwise $2^{\ell_a}/a < 2$ restarts on average

Secure Modulo Reduction: Protocol

- Input: $[\![x]\!], a$, with $x < 2^{\ell_x}$ and $2^{\ell_a - 1} < a \leq 2^{\ell_a}$
- Requirement: $an2^{\ell_x + \ell_s} < N^s$ for security parameter ℓ_s
- For computing $[\![x \bmod a]\!]$, the n servers do:
  1. Jointly construct $[\![r]\!]^{b(\ell_a)}$ for $r \in_R [0, a)$
  2. Individually construct $[\![s_i]\!]$ for $s_i \in_R \{0, 1\}^{\ell_x + \ell_s}$
  3. Individually compute $[\![\tilde{x}]\!] = [\![x]\!][\![r]\!]^{-1} \prod_{i=1}^{n} [\![s_i]\!]^a$ (note that $\tilde{x} = x - r + a \sum_{i=1}^{n} s_i$ and $0 \leq \tilde{x} < N^s$)
  4. Jointly decrypt $[\![\tilde{x}]\!]$ and compute $\bar{x} = \tilde{x} \bmod a \equiv x - r \bmod a$ (note that $\bar{x} + r \equiv x \bmod a$ and $0 \leq \bar{x} + r < 2a$)
  5. Using comparison gate, compute $[\![c]\!] = [\![[a - 1 - \bar{x} < r]]\!]$ (note that $c = 0 \iff \bar{x} + r < a$)
  6. Individually compute output $[\![\bar{x}]\!][\![r]\!][\![c]\!]^{-a}$
- Protocol can be simulated in framework of [CDN01]

Secure Modulo Reduction: Security Proof

- Simulated for $[\![x]\!] = [\![x^{(0)}(1-b) + x^{(1)}b]\!]$ given $x^{(0)}, x^{(1)}, [\![b]\!]$
    - Distinguisher for simulator is a distinguisher for bit-decryption
- n participants $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ of which $\{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\}$ are malicious
    1. Simulator takes $r \in_R [0, a)$, but simulates this phase with $\tilde{r} = \tilde{r}^{(0)}(1-b) + \tilde{r}^{(1)}b$, where $\tilde{r}^{(b)} = \left(r + x^{(b)}\right) \bmod a$
    2. Simulator lets the malicious parties construct and prove s_i. For $\mathcal{P}_t, \ldots, \mathcal{P}_{n-1}$ he executes the protocol as is. For $\mathcal{P}_n$ he takes $s_n \in_R [0, 2^{\ell_x + \ell_s})$, but simulates with $[\![\tilde{s}_n]\!] = [\![\tilde{s}_n^{(0)}(1-b) + \tilde{s}_n^{(1)}b]\!]$, where $\tilde{s}_n^{(b)} = s_n - \left(r + x^{(b)}\right) \operatorname{div} a$
    3. Executes this phase. He obtains $[\![\tilde{x}]\!] = [\![x - \tilde{r} + a\sum_{i=1}^{n-1} s_i + a\tilde{s}_n]\!] = [\![-r + a\sum_{i=1}^{n} s_i]\!]$
    4. Simulates the decryption on input $[\![\tilde{x}]\!]$ and $-r + a\sum_{i=1}^{n} s_i$
    5. Comparison gate is simulated
- $\tilde{r}$ and $\tilde{s}_n$ indistinguishable from r and s_n