Evaluation of **HIMMO with Long Identifiers**

an Extension of the HIMMO Key Establishment Scheme

Cristian-Alexandru STAICU

EIT ICT Labs Master School, University of Twente, University of Trento
University Supervisor: Dr. Andreas Peter
Industry Supervisor: Dr. Oscar Garcia-Morchon

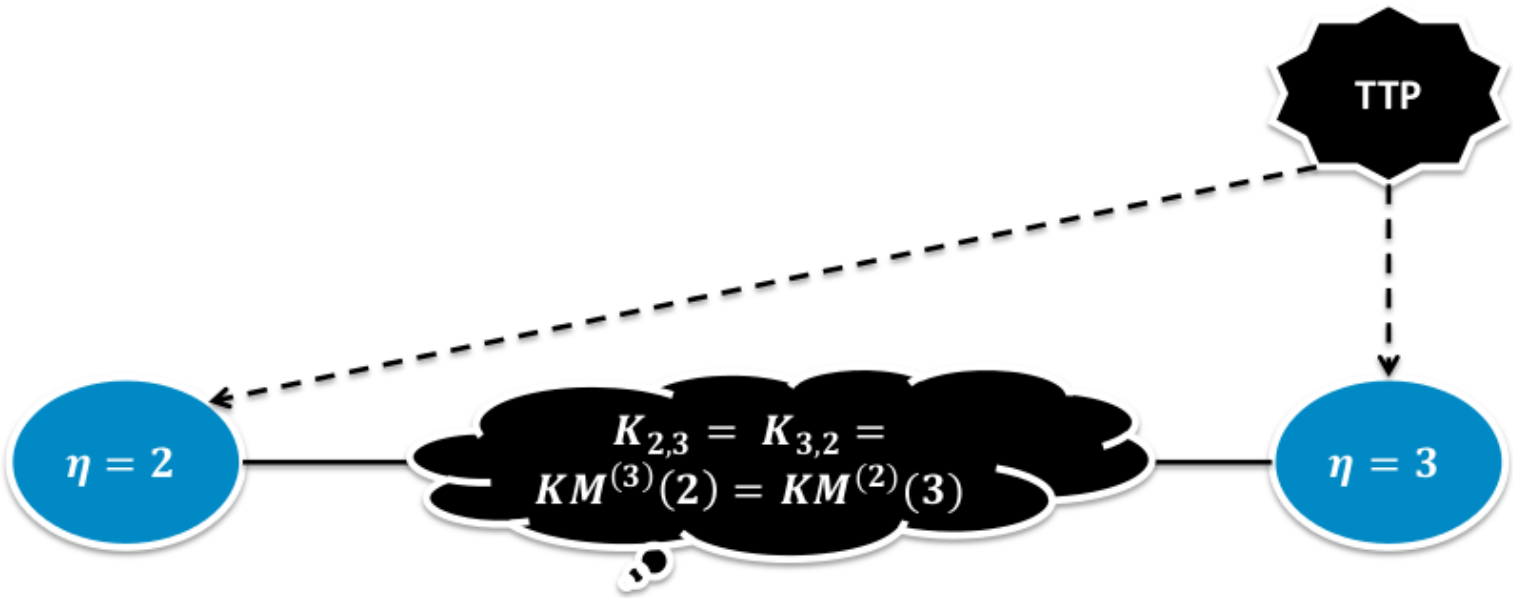
28th August 2014

Overview

Key Establishment / Key Predistribution

Key Establishment

Key establishment is any process whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic use. *(Handbook of Applied Cryptography)*



[1]

Figure : Key Predistribution Scheme

[1] TTP = Trusted Third Party

Polynomial Schemes

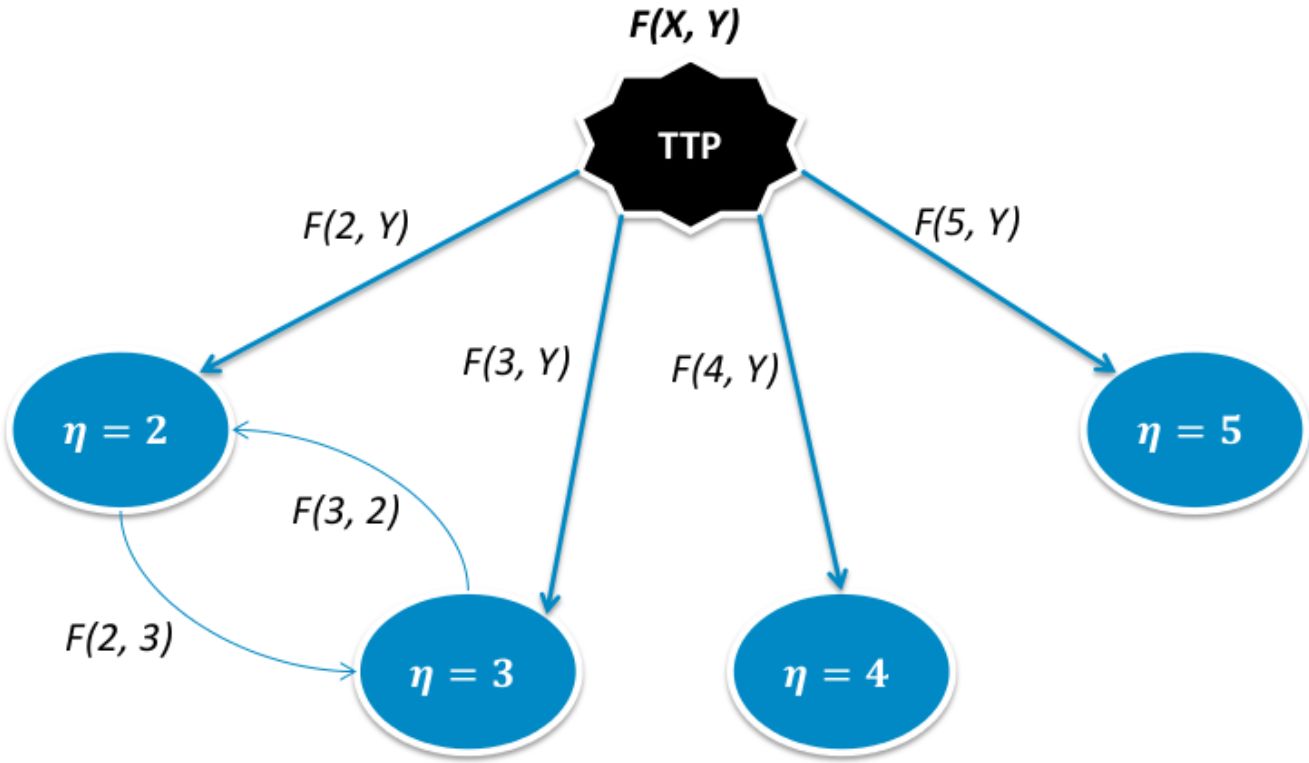


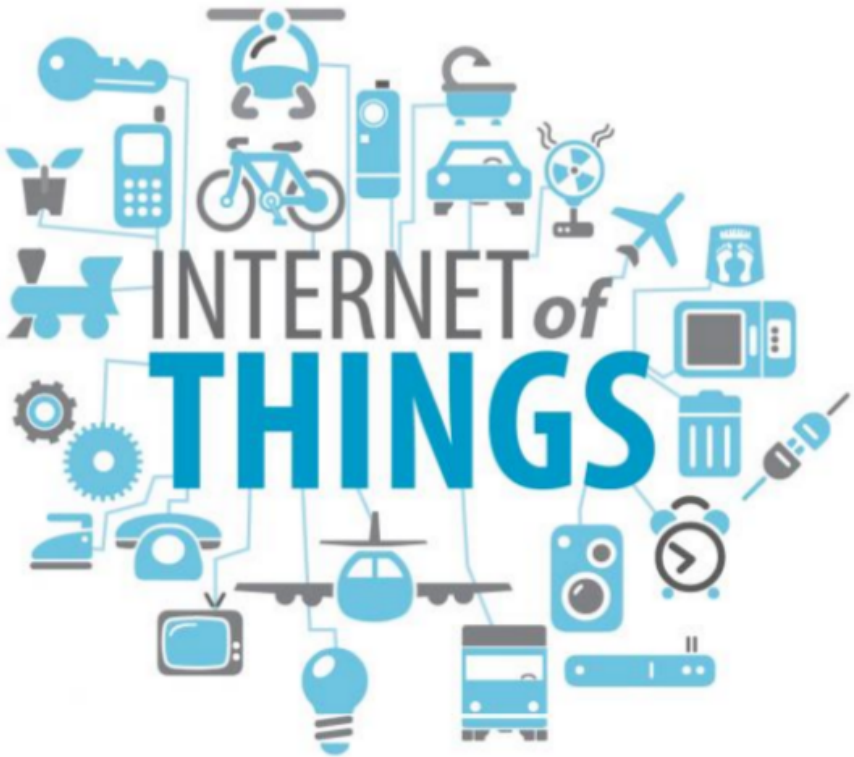
Figure : Blundo's Scheme

Observations

$F(X, Y) = F(Y, X)$

Interpolation: $\alpha + 1$ colluding nodes can retrieve F

Internet of Things



Properties

- lightweight devices,
- real time,
- high number of devices,
- multi-interface,
- multimodal interaction,
- huge amount of data,
- limited bandwidth,
- context aware,
- security and privacy issues.

HIMMO Key Establishment Scheme

Hide Information Problem

Let $f(X) \in \mathbb{Z}_N[X]$ be a polynomial of degree at most α and let $f(\xi)_b = \langle \langle f_b(\xi) \rangle_N \rangle_{2^b}$ be the last b bits of the evaluated polynomial modulo N. The HI problem is to construct a polynomial time algorithm that can reconstruct f based on $k \geq 1$ different pairs $(\xi, f_b(\xi))$.

Mixing Modular Operations Problem

Let $m, q_1, q_2, ..., q_m, N$ be some positive integers, $f_1, f_2, ..., f_m$ be polynomials of degree at most $\alpha \geq 2$. Having $h(\xi) = \left\langle \sum_{i=1}^{m} \langle f_i(\xi) \rangle_{q_i} \right\rangle_N \ \forall \xi \in \mathbb{Z}$, the MMO problem is to construct a polynomial time algorithm that recovers $f_1, f_2, ..., f_m$ given m, α, N and $k \geq 1$ pairs $(\xi_i, h(\xi_i))$

Problem Statement

- Can we increase security in other way then by using larger degree polynomials?
- Can we use multiple parallel systems with small key size?
- How well does HIMMO perform (computation time, memory?
- How well does it perform compared with other schemes?

Contributions

- Described HIMMO with long identifiers (HIMMO-LI), an extension that aims at higher security and scalability. Designed algorithms for HIMMO-LI.
- Implemented and evaluate the designed algorithms on a 8-bit AVR microcontroller.
- Compared HIMMO with other key establishment schemes.

Setup and Node Registration in HIMMO-LI

Initialization Phase

- TTP se key size (b), identifiers size (B), public modulus (N) and m moduli (q_i).
- TTP selects m symmetric bivariate polynomials $f_i \in \mathbb{Z}_{q_i}[X, Y]$ of degree α.
- N, q_i are $(\alpha + 1)B + b$ bits long numbers

Registration Phase

TTP computes and delivers to each node η:

$$\mathrm{KM}_{\eta,j} = \left\langle \sum_{i=1}^{m} \langle f_{i,j}(\eta) \rangle_{q_i} \right\rangle_N$$

Operational Phase in HIMMO-LI

Operational Phase

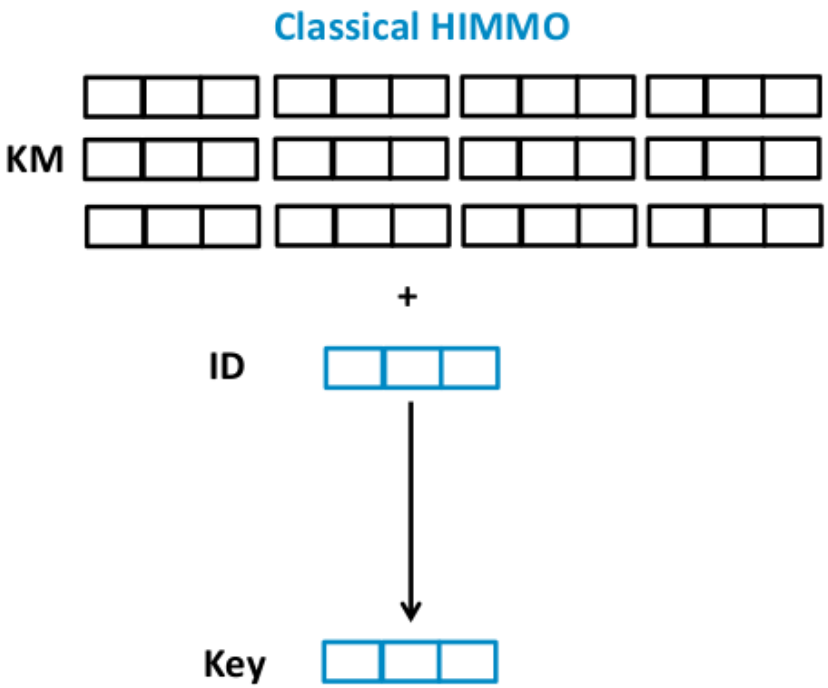
- The difference from Blundos: only the last b bits are used as key.
- Every two nodes can compute a shared key:

$$K_{i,j} = \left\langle \left\langle \sum_{k=0}^{\alpha} \mathrm{KM}_{i,k}(\xi_j)^k \right\rangle_N \right\rangle_{2^b} .$$

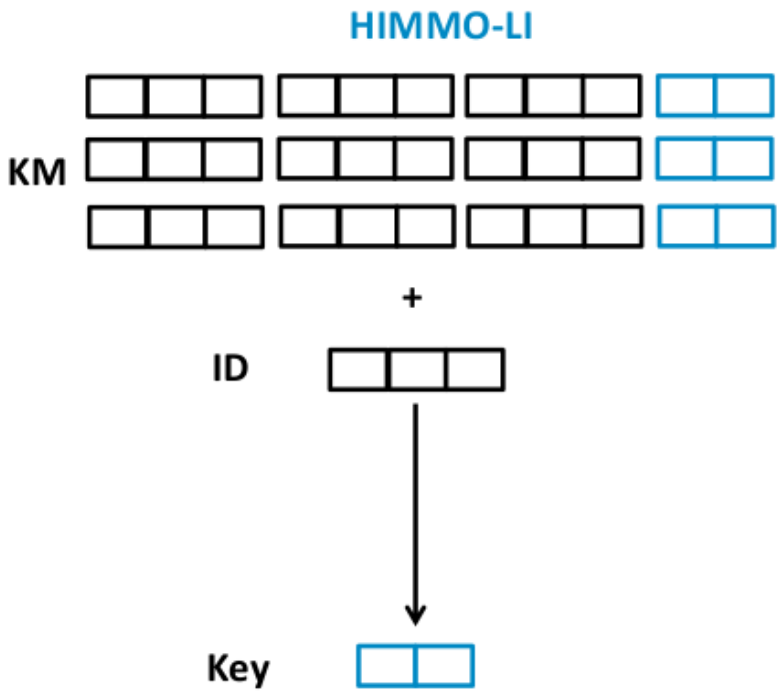
- The computed keys are approximately equal:

$$K_{i,j} \in \{\langle K_{j,i} + kN \rangle_{2^b} | -\Delta \leq k \leq \Delta\},\ \Delta = 3m + \alpha + 1.$$

HIMMO vs. HIMMO-LI

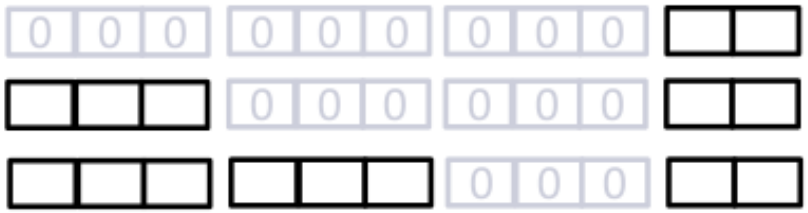


$\alpha = 2, m = 2, b = 3$



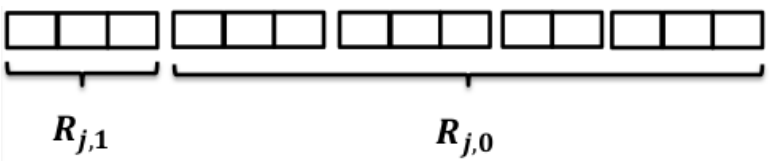
$\alpha = 2, m = 2, B = 3, b = 2$

More is Less



$$N = 2^{(\alpha+1)B+b} - 1$$

$$\left\langle 2^{(\alpha+1)B+b} \right\rangle_N = 1$$

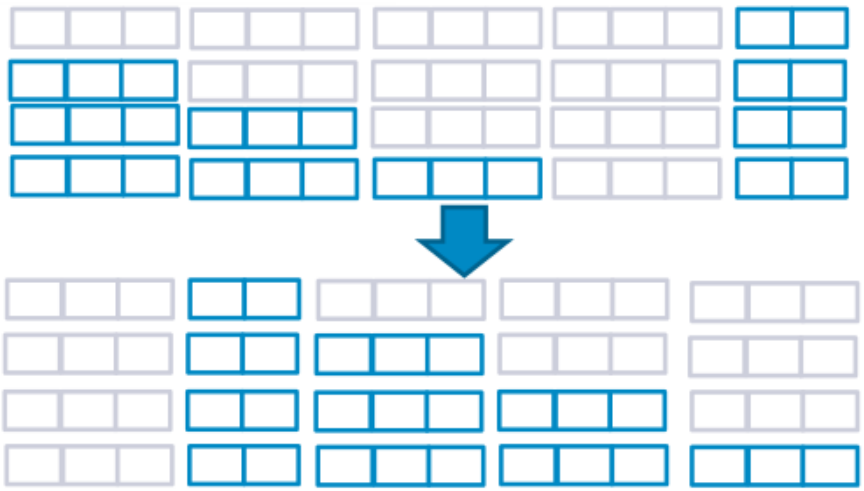


$$\langle R_j \rangle_N = R_{j,1} + R_{j,0}$$

Data: $\alpha, b, B, \eta', \mathrm{KM}_{\eta,j}$ where $j \in 0, 1, ..., \alpha$
Result: $\langle \langle \sum_{j=0}^{\alpha} \mathrm{KM}_{\eta,j}\, \eta'^{j} \rangle_N \rangle_{2^b}$

$key = \langle \mathrm{KM}_{\eta,\alpha} \rangle_{2^b}$
$temp = \langle \mathrm{KM}_{\eta,\alpha} \rangle_{2^{(\alpha+2)B}}$
for $j = \alpha \rightarrow 1$ **do**
  if $j = \alpha - 1$ **then**
    $temp = (temp \gg b) \cdot \eta'$
  else
    $temp = (temp \gg B) \cdot \eta'$
  end
  $temp = \langle temp \rangle_{2^{(j+2)B}}$
  $temp = temp + \mathrm{KM}_{\eta,j} \gg b$
  $key = \langle key \cdot \eta' \rangle_{2^b}$
  $key = \langle key + \langle \mathrm{KM}_{\eta,j} \rangle_{2^b} \rangle_{2^b}$
  $key = key + temp \gg (j+1)B$
end

HIMMO Transformation



Data: $\alpha, b, B, \eta', \mathrm{KM'}_{\eta,j}$ where $j \in 0, 1, ..., \alpha$
Result: $\langle \langle \sum_{j=0}^{\alpha} \mathrm{KM}_{\eta,j}\, \eta'^{j} \rangle_N / 2^{\alpha B} \rangle_{2^b}$

1 $key = \mathrm{KM'}_{\eta,\alpha}$
2 **for** $j = \alpha \rightarrow 1$ **do**
3 $\quad key = key * \eta'$
4 $\quad key = key \gg B$
5 $\quad key = key + \mathrm{KM'}_{\eta,j}$
6 **end**
7 $key = key \gg B$

TTP Transforms the Key Material

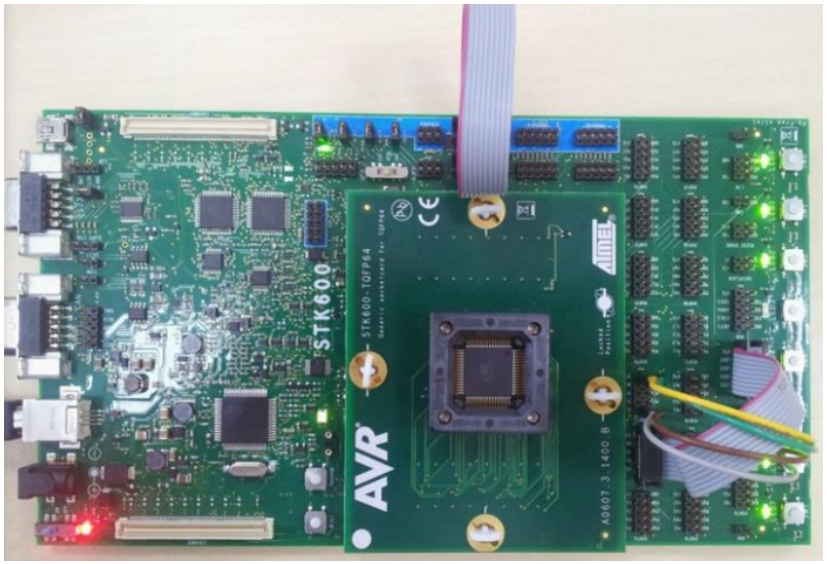
$$\mathrm{KM}'_j = \left\langle \mathrm{KM}_j * 2^{\alpha B} \right\rangle_N$$

Theoretical Analysis for the Algorithms

Parameter	More is Less	HIMMO Transformation Performance Gain
Time	$\mathcal{O}\left(\frac{n(\alpha^2 B^2+\alpha b^2)}{\mathrm{CPUW}^2}\right)$	$\mathcal{O}\left(\frac{n(\alpha B^2+\alpha b^2)}{\mathrm{CPUW}^2}\right)$
FLASH	$\mathcal{O}\left(\frac{n(\alpha^2 B+\alpha b)}{\mathrm{CPUW}}\right)$	$\mathcal{O}\left(\frac{n\alpha B}{\mathrm{CPUW}}\right)$
RAM	$\mathcal{O}\left(\frac{n(\alpha B+b)}{\mathrm{CPUW}}\right)$	0

STK600 / Atmega 128L Implementation

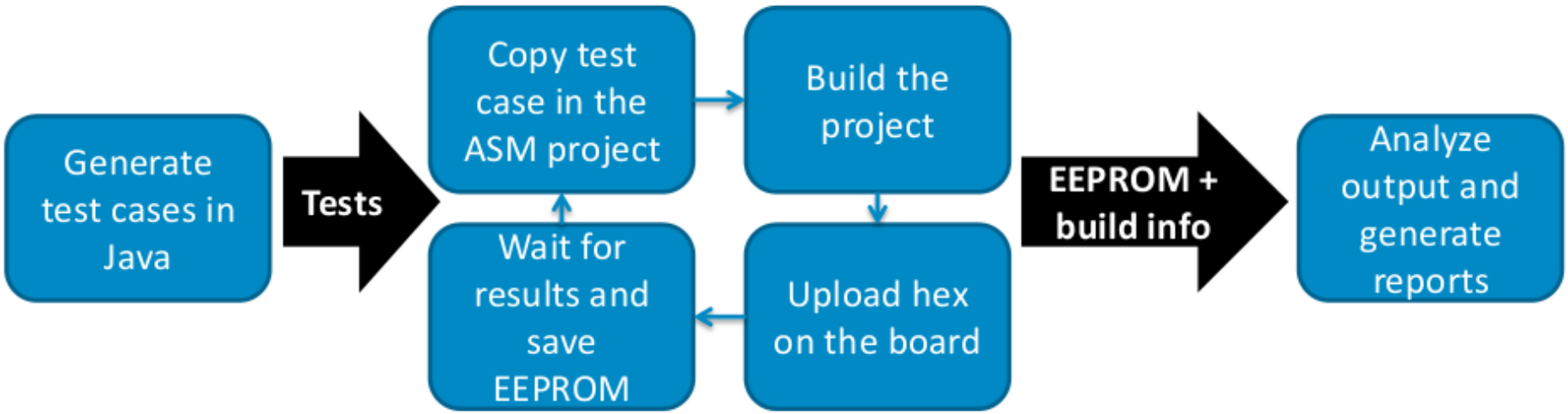
- 8 bit RISC architecture with 32 registers,
- 128K FLASH, 4K EEPROM, 4K RAM,
- AVR Dragon board on JTAG interface for debugging,
- multi-precision arithmetic to emulate long number operations,
- use EEPROM memory for output,
- use interrupts to handle the timer's overflow.



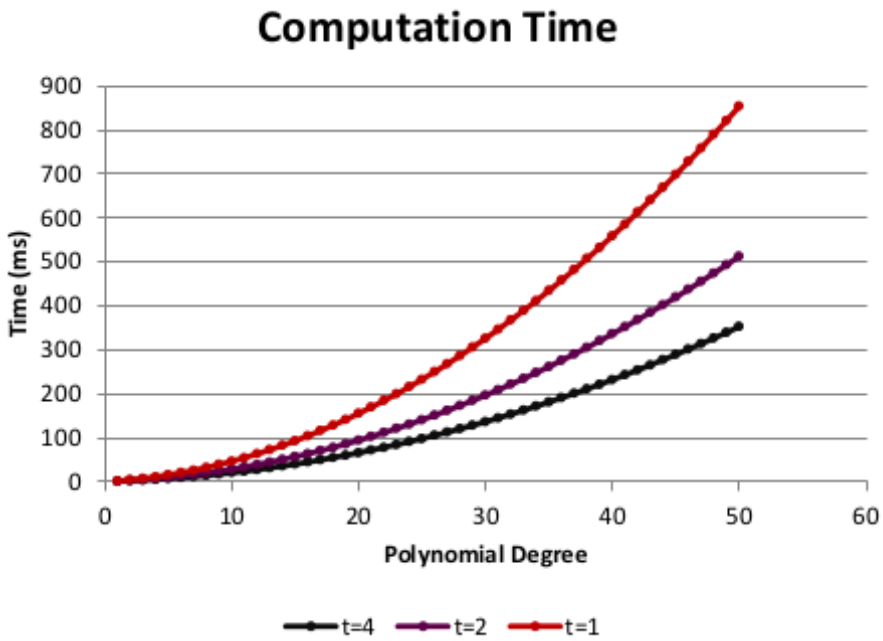
Parameters to be Evaluated

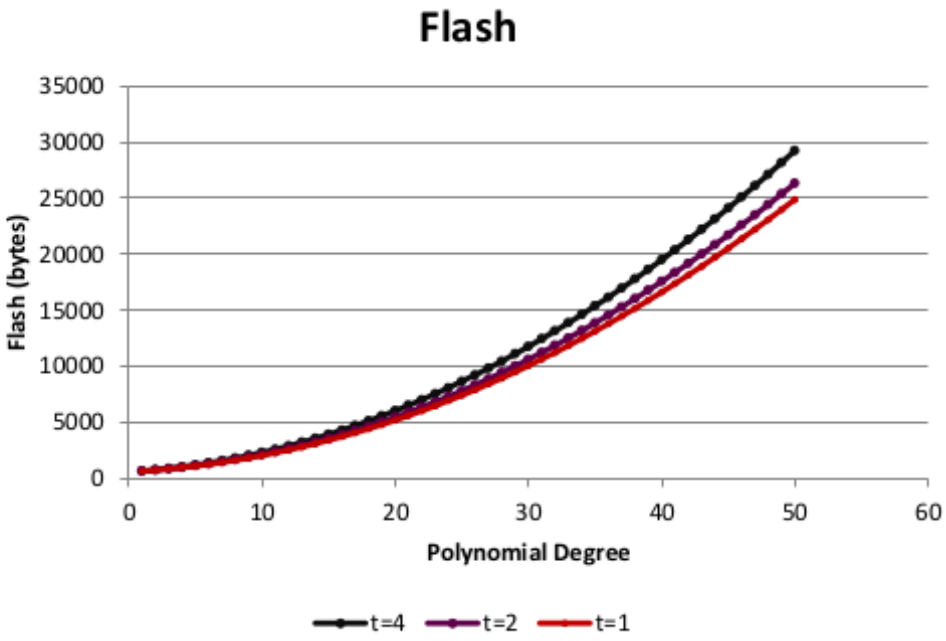
- number of cycles / CPU time measure using hardware counter TCNT1,
- flash memory (code size + key material) measure using the IDE static output,
- RAM memory measure using the IDE static output,
- quality of the implementation / correctness of the computed key,
- number of HIMMO instances needed to compute a key of certain length.

Testing and Data Collection Framework

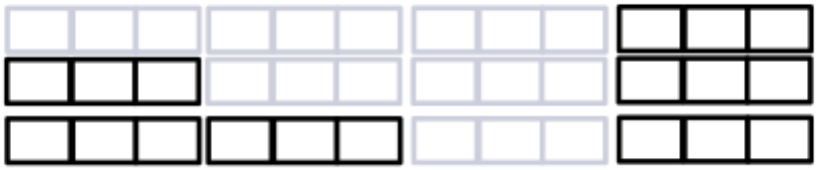


Computing 128-bit key





Collusion Attacks. Security Properties



$$c = \frac{(\alpha+1)(\alpha+2)}{2}$$

$$c = \frac{B}{b}\frac{\alpha(\alpha+1)}{2} + \alpha + 1$$

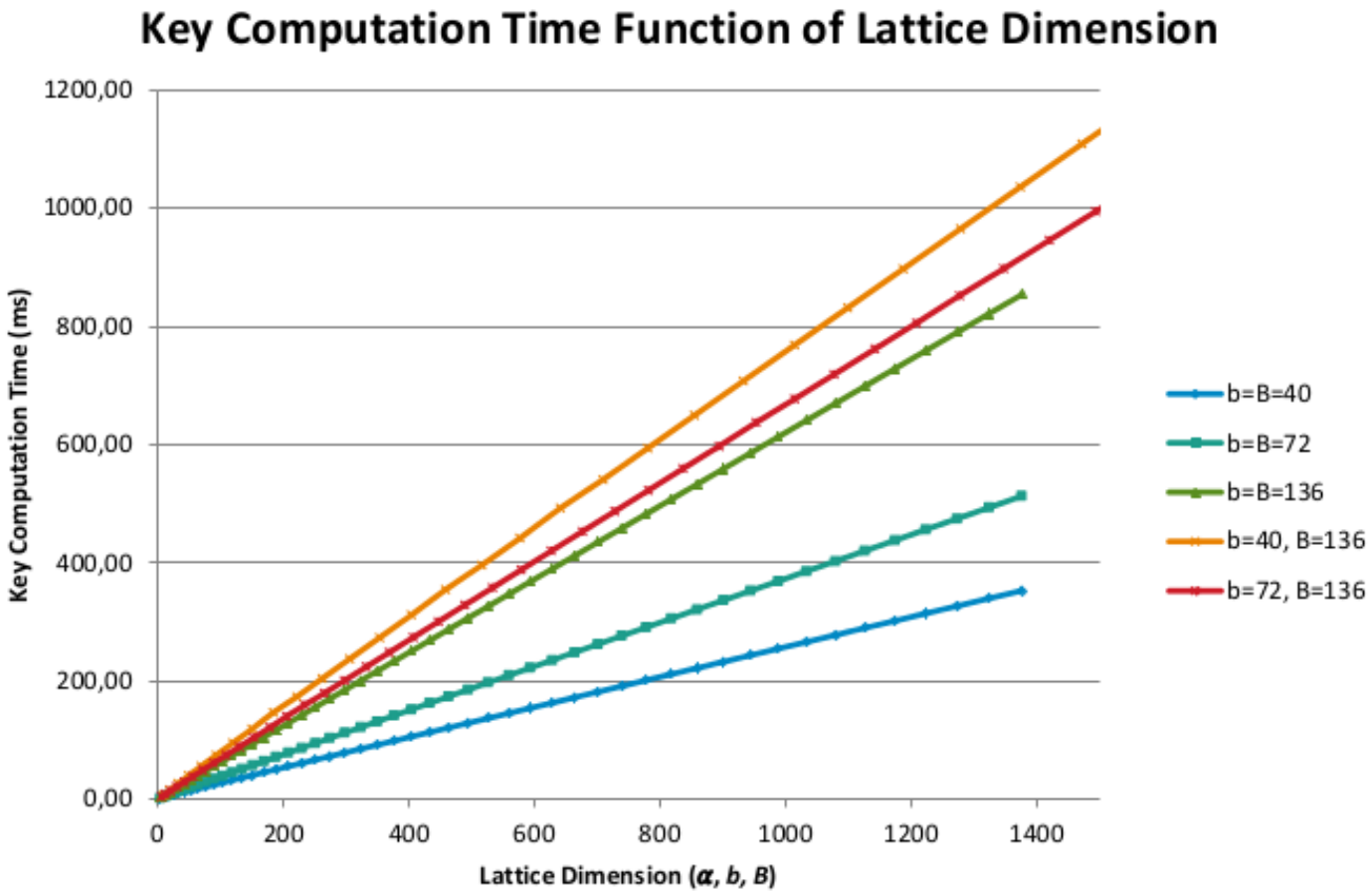
Lattice Dimension

$$\mathrm{ld} = c + \alpha + 1$$

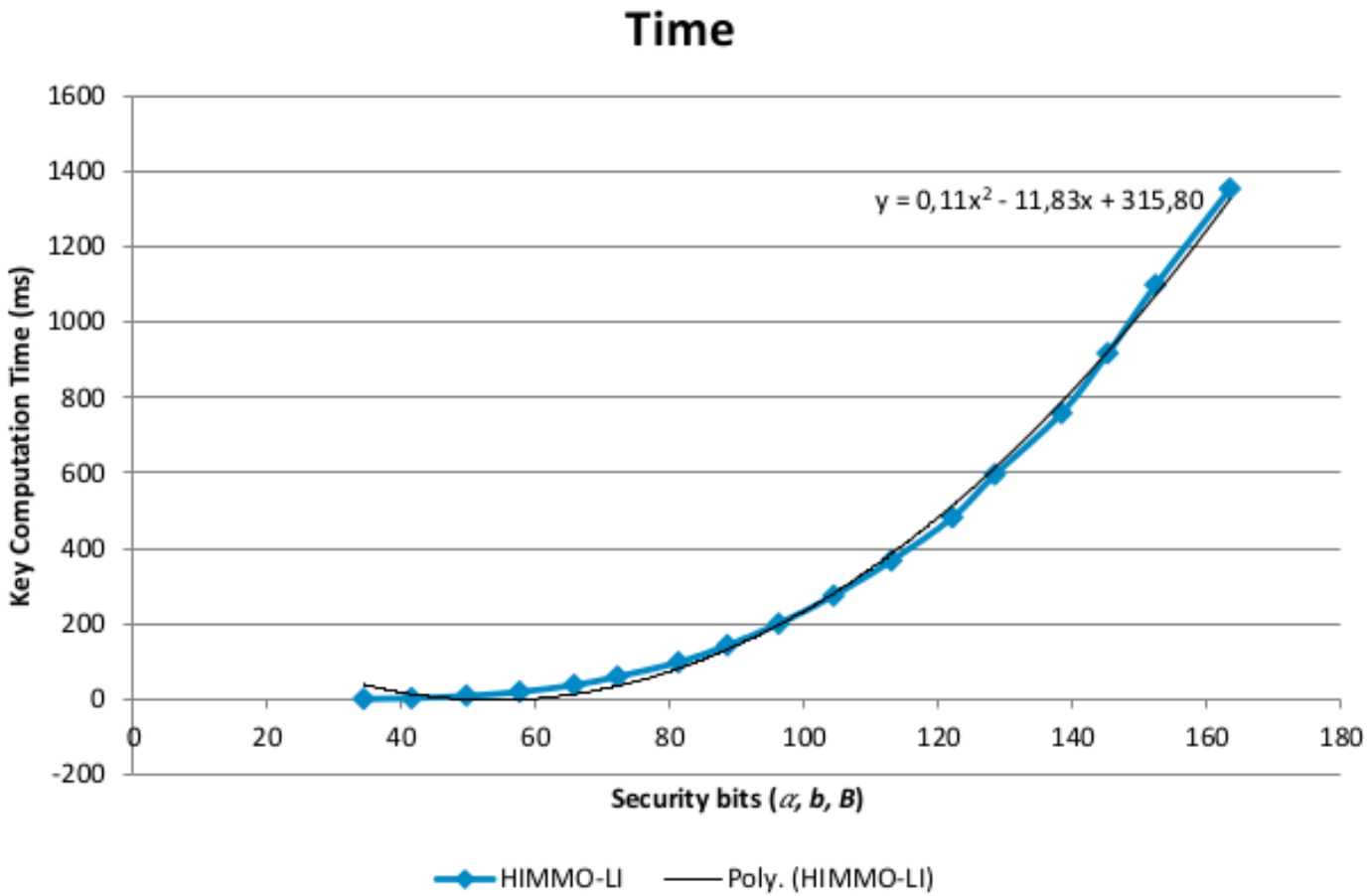
Bits of Security

$$\mathrm{sb} = f(\mathrm{ld}, \mathrm{elsize})$$

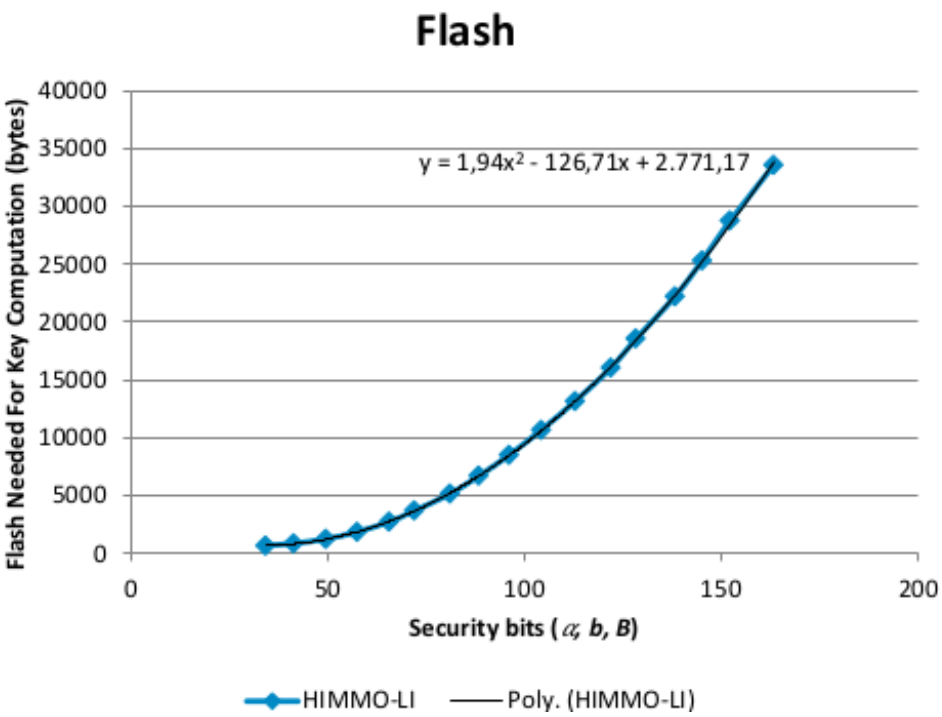
Achieving Certain Levels of Security using Different Instances

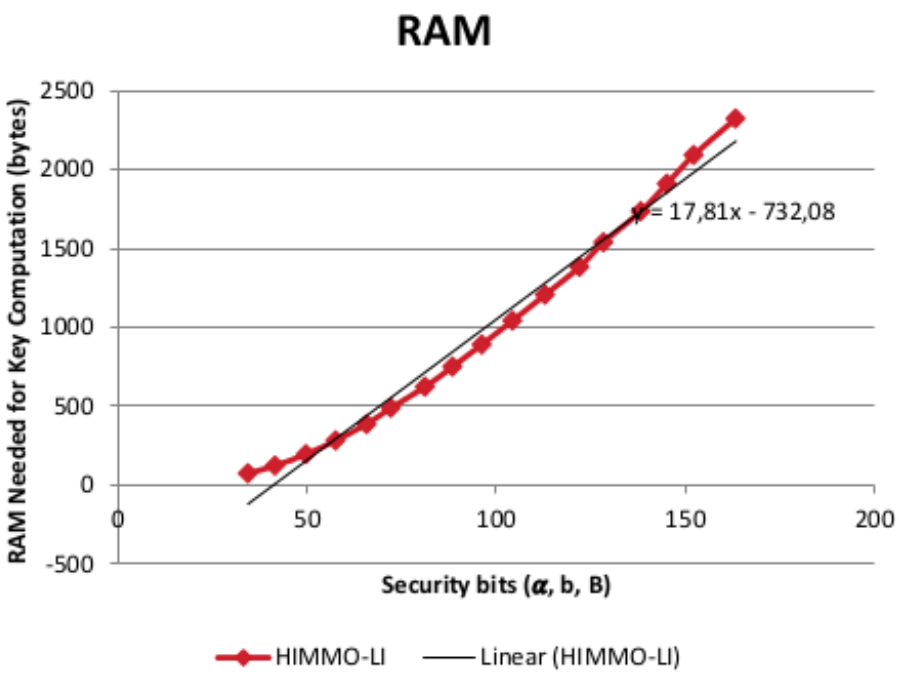


Computation Time for Security Bits



Memory for Security Bits





HIMMO-LI vs. Other Schemes for 80 Bits of Security

Protocol	**Bandwidth**	**Round Trips**	**CPU Time**	**Security Properties**
ECDH	480	1	3960.70	key confirmation, forward secrecy
ECDH-ECDSA	704	1	11901.92	key confirmation, authentication, forward secrecy
HIMMO-AES	**320**	0/1	**202.01**	key confirmation, authentication
IBE	700+key size	0/1	>6213.6	key confirmation, authentication, forward secrecy

Conclusions

- Designed algorithms for the More is Less and HIMMO Transformatio,n optimizations of the HIMMO scheme, to include the long identifiers extension.
- Implemented them in C and ASM for ATmega128 microprocessor, but also in Java.
- Observed certain parameters of the obtained implementations and automatized the process.
- Compared the results with the ones in the literature for similar key agreement schemes.
- Run more than 10000 tests on the ASM implementation, hinting on its correctness