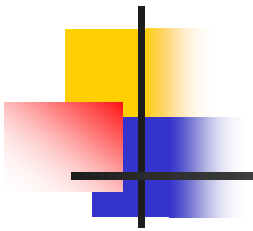
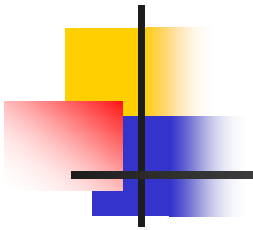


Criptografia em RSSF: implementações e impactos nos nós sensores

Apresentação das implementações de criptografia em RSSF e o impacto no consumo de energia e na ocupação de memória em nós sensores

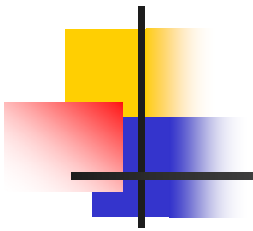
Tópicos:

- **Revisão de criptografia**
- **Distribuição de chaves criptográficas**
- **Criptografia no TinyOS: TinySec**
- **Distribuição de chaves em RSSF**
- **Conclusões**



Criptografia em RSSF: implementações e impactos nos nós sensores

Revisão de criptografia



Criptografia em RSSF: implementações e impactos nos nós sensores

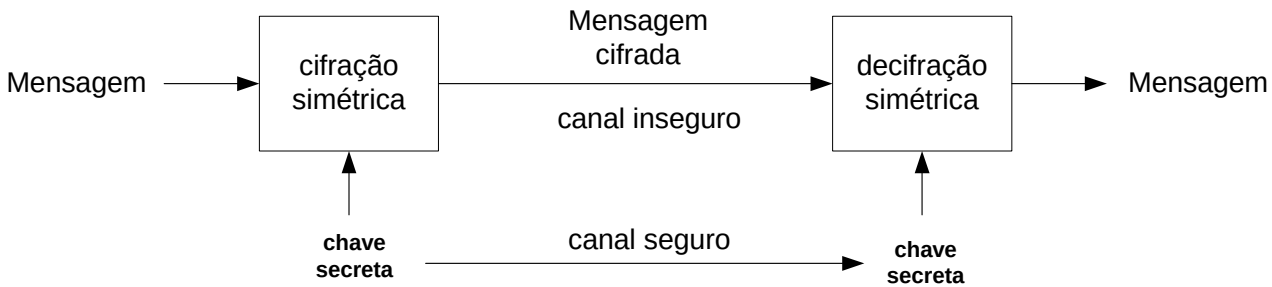
Classificação dos algoritmos:

Simétricos

Assimétricos

Criptografia em RSSF: implementações e impactos nos nós sensores

Criptografia com algoritmos simétricos (chave secreta)

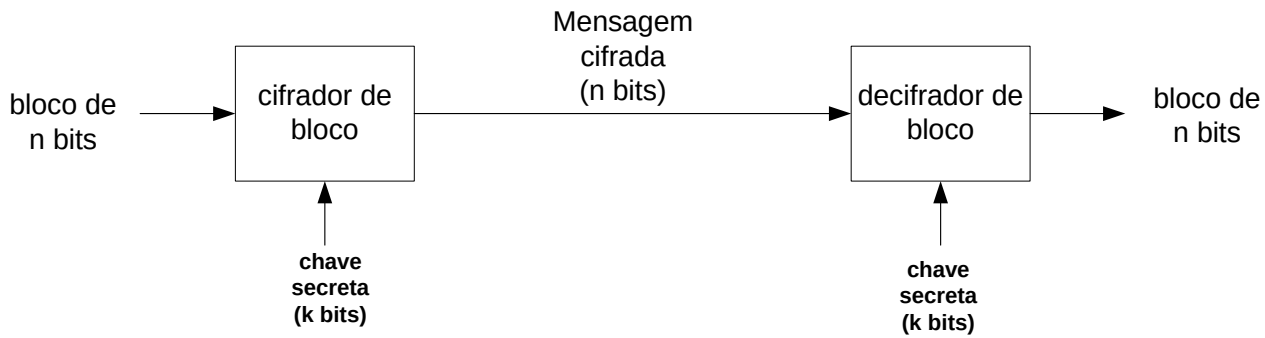


Tipos de algoritmos simétricos:

Algoritmos de bloco
Algoritmos de fluxo

Criptografia em RSSF: implementações e impactos nos nós sensores

Algoritmos de bloco (cifração de blocos de n bits)



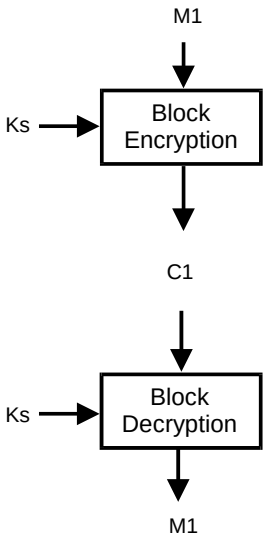
Utilizam substituições e transposições: implementação simples e processamento rápido.

Utilização: Confidencialidade

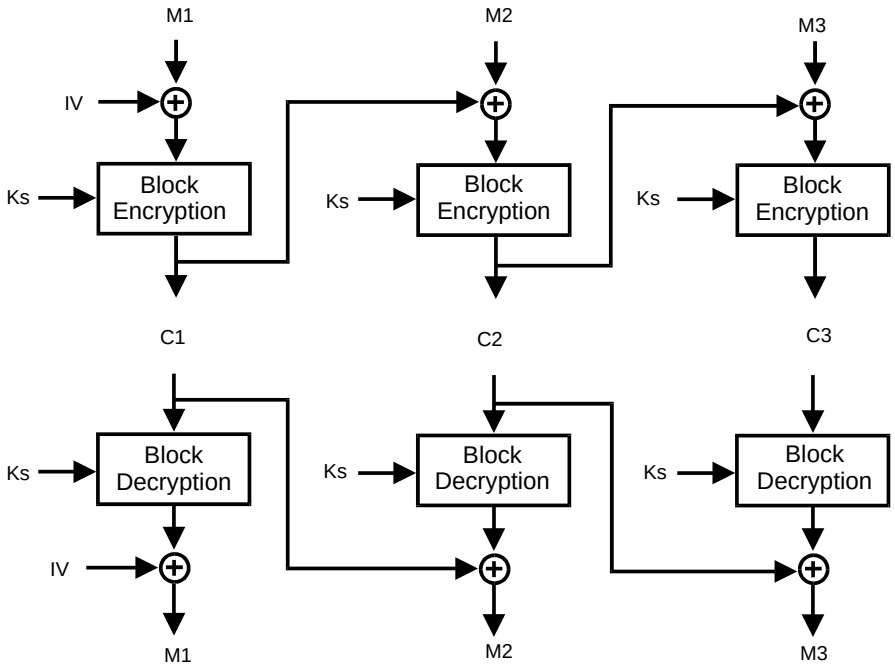
Criptografia on-line.

Criptografia em RSSF: implementações e impactos nos nós sensores

Modos de cifração de bloco



ECB mode



CBC mode

Criptografia em RSSF: implementações e impactos nos nós sensores

Algoritmos mais utilizados:

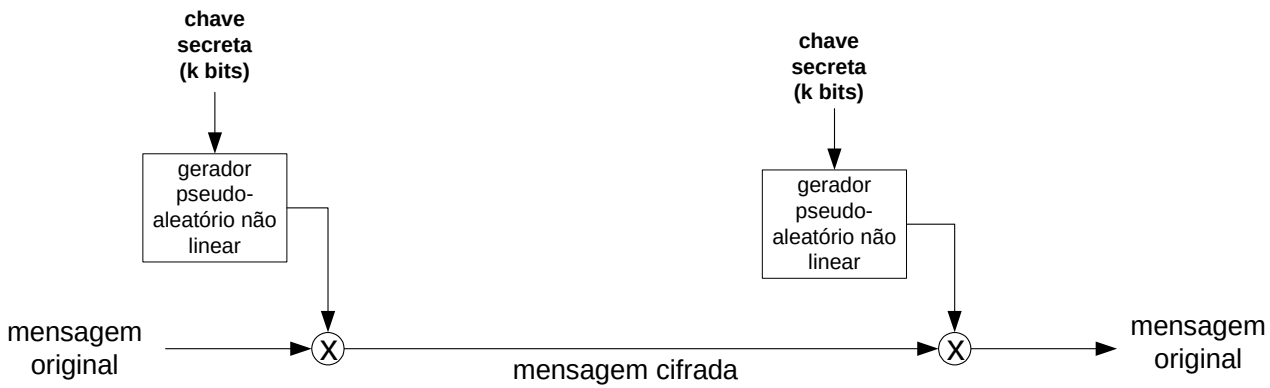
DES: n = 64 bits, k = 56 bits
3DES: n = 64 bits, k = 168 bits
IDEA: n= 64 bits, k = 128 bits
AES: n = 128 bits, k = 128, 192, 256 bits

Os mais adequados para RSSF:

Skipjack: n = 64 bits, k = 80 bits (projetado para uso em chips
RC5: n e k podem ser selecionados (o mais adequado – alto desempenho e pouco espaço em memória)

Criptografia em RSSF: implementações e impactos nos nós sensores

Algoritmos de fluxo (cifração do fluxo de bits)



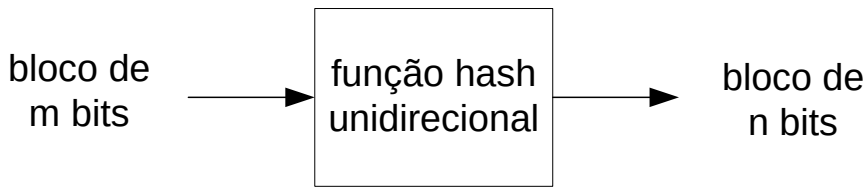
Exemplo: RC4 (k de 1 a 2048 bits, seqüência maior que 10^{100})

Utilização: Confidencialidade

Criptografia on-line.

Criptografia em RSSF: implementações e impactos nos nós sensores

Função resumo criptográfica (função hash unidirecional)



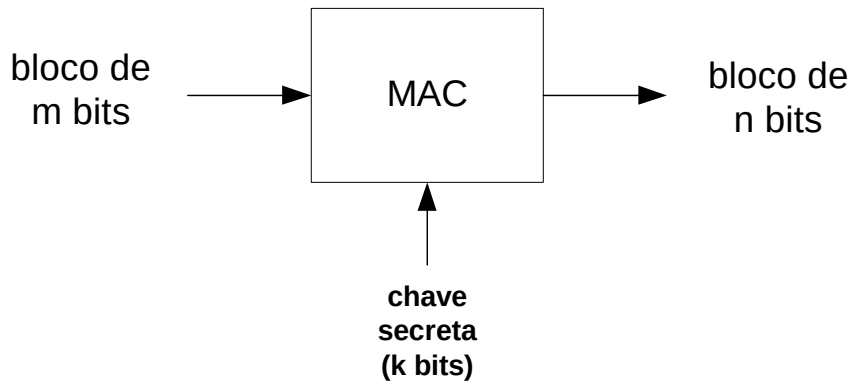
Exemplos:

MD5: m = 512 bits, n = 128 bits
SHA-1: m = 512 bits, n = 160 bits

Criptografia em RSSF: implementações e impactos nos nós sensores

Message Autentication Code (MAC)

Função hash unidirecional com chave:

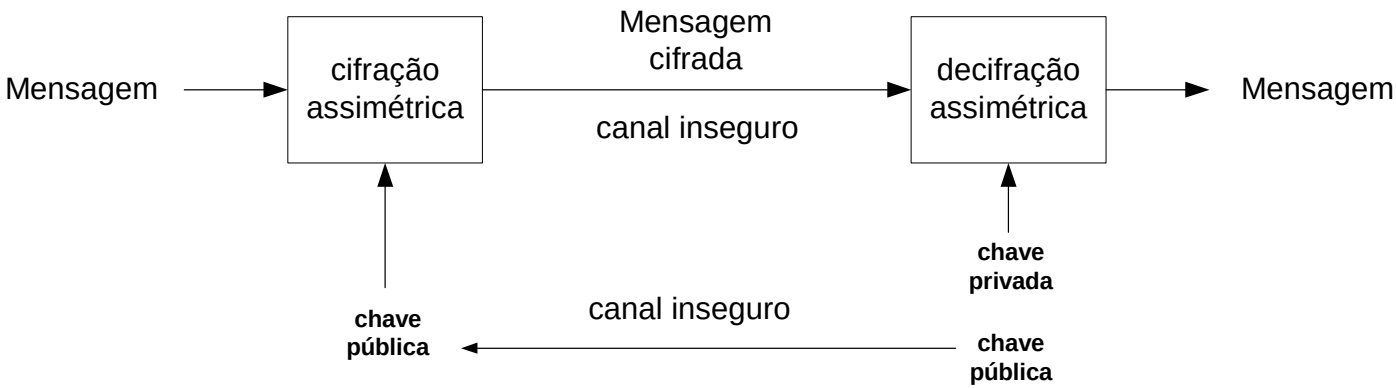


Aplicação: autenticação e integridade

Criptografia em RSSF: implementações e impactos nos nós sensores

Criptografia com algoritmos assimétricos (chave pública)

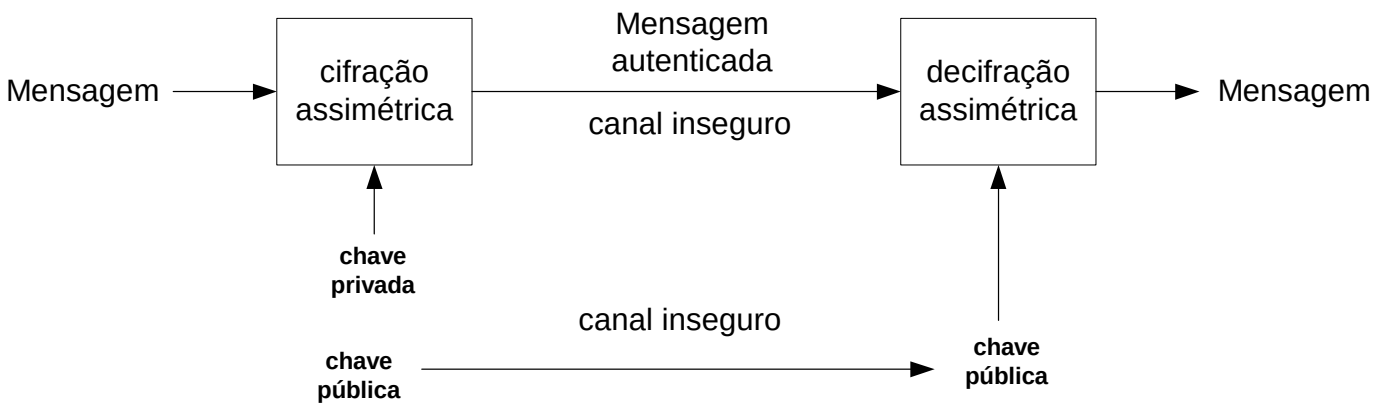
Aplicação: Confidencialidade e distribuição de chaves secretas



Criptografia em RSSF: implementações e impactos nos nós sensores

Criptografia com algoritmos assimétricos (chave pública)

Aplicação: autenticação



Criptografia em RSSF: implementações e impactos nos nós sensores

Criptografia com algoritmos assimétricos (chave pública)

Utilizam operações matemáticas complexas: implementação complexa e processamento lento.

Exemplos: RSA (fatoração de grandes números), ElGamal (logaritmo discreto em campos finitos)

Criptografia em RSSF: implementações e impactos nos nós sensores

Algoritmo RSA (exponenciação módulo n)

Dificuldade de fatoração de grandes números

- selecionar $n = p.q$ onde p e q são primos distintos (um dos dois deve ter de 1024 a 2048 bits)
- selecionar $e < (p-1).(q-1)$ tal que e e $(p-1).(q-1)$ sejam primos entre si
- calcular $d = e^{-1} \bmod (p-1).(q-1)$

Chave pública: (n, e), chave secreta (d)
Mensagem: m

Cifração:

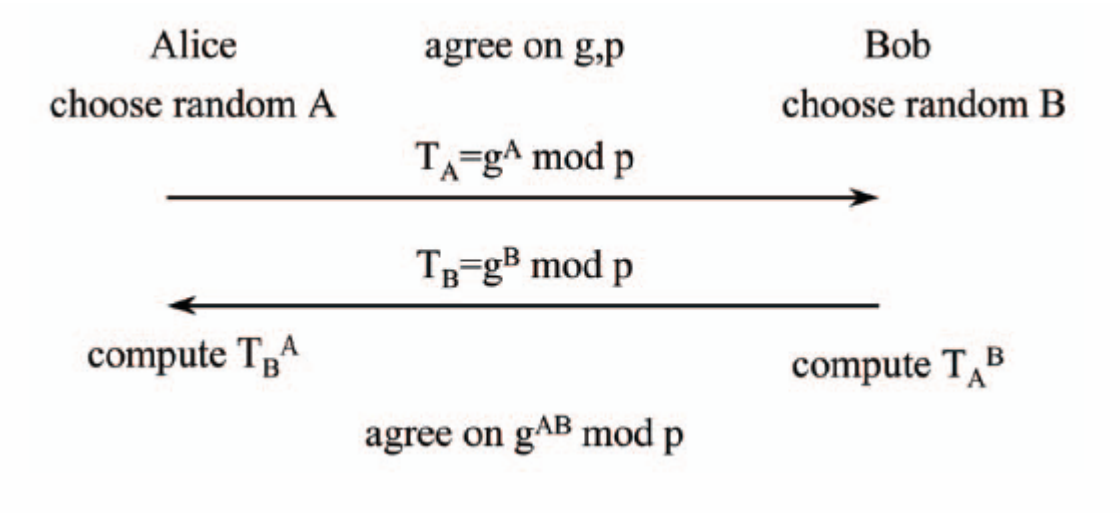
$c = m^e \bmod n$ (cifração)

Decifração:

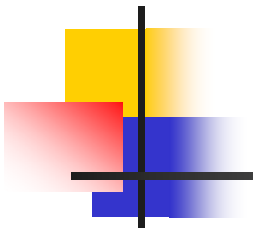
$m = c^d \bmod n$ (decifração)

Criptografia em RSSF: implementações e impactos nos nós sensores

Diffie-Hellman (DLP – Discrete Logarithm Problem)



Utilização: distribuição de chaves



Criptografia em RSSF: implementações e impactos nos nós sensores

Distribuição de chaves criptográficas

Criptografia em RSSF: implementações e impactos nos nós sensores

Segurança criptográfica depende da chave secreta nos algoritmos bem projetados:

- Distribuição segura de chaves é fundamental;
- Uma chave descoberta permite decifração de todos os dados enviados antes e depois.

Técnicas de criptoanálise:

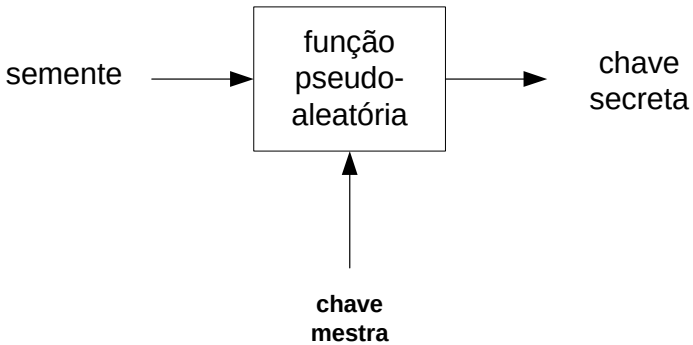
- Troca de chave dificulta muito a criptoanálise.

Métodos para distribuição de chaves:

- Chave mestra;
- Algoritmos assimétricos.

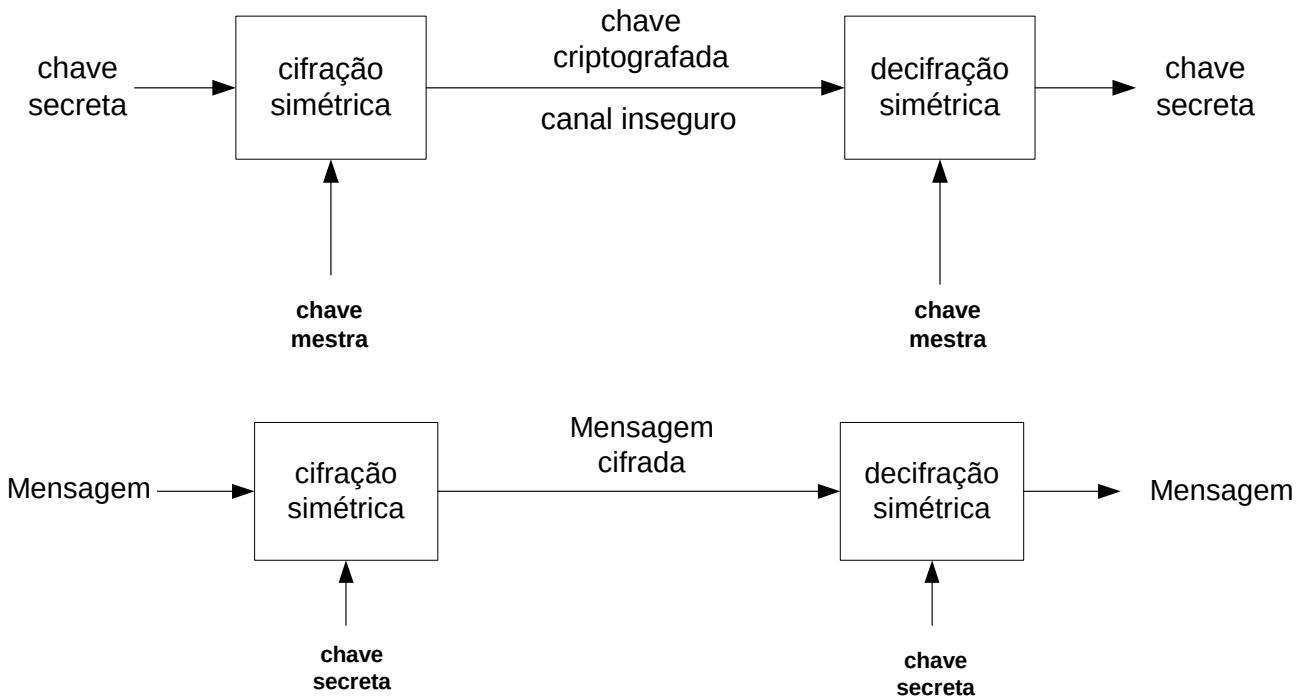
Criptografia em RSSF: implementações e impactos nos nós sensores

Distribuição com Chave mestra (sem comunicação)



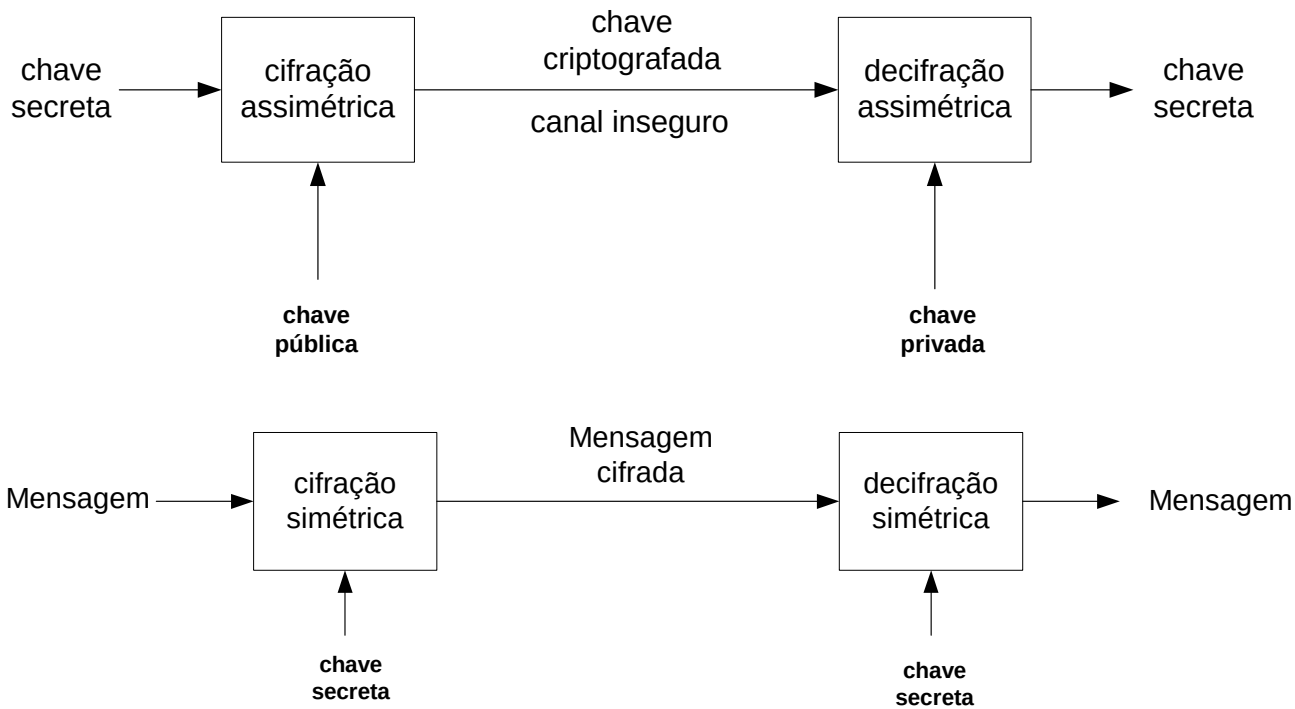
Criptografia em RSSF: implementações e impactos nos nós sensores

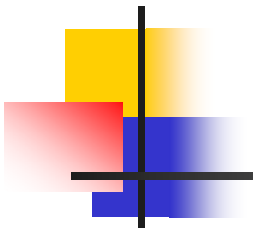
Distribuição com Chave mestra (com comunicação)



Criptografia em RSSF: implementações e impactos nos nós sensores

Distribuição com Algoritmos assimétricos





Criptografia em RSSF: implementações e impactos nos nós sensores

Criptografia no TinyOS: TinySec

Criptografia em RSSF: implementações e impactos nos nós sensores

TinySec

Segurança no nível de link (chave secreta fixa)

Novo componente:

SecureGenericComm em substuição ao GenericComm

Interfaces:

SendMsg: Integridade, autenticação, sem criptografia

SendMsgEncryptAndAuth: Integridade, autenticação, com criptografia

SendMsgCRC: envia mensagem com CRC, sem criptografia

ReceiveMsg: recebe mensagens

ReceiveMsgNoAuth: recebe mensagens com CRC

Algoritmos de bloco:

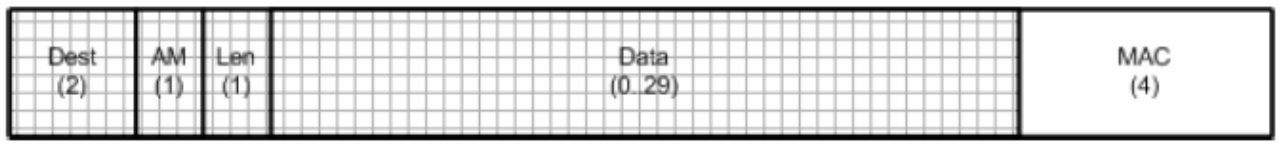
RC5 (12 etapas, blocos de 64 bits, chave de 80 bits)

Skipjack

Criptografia em RSSF: implementações e impactos nos nós sensores

Formato dos pacotes do TinySec

(a) TinySec-AE packet format



(b) TinySec-Auth packet format

(c) TinyOS packet format

Criptografia em RSSF: implementações e impactos nos nós sensores

Impacto da criptografia no Mica2 (TinySec)

Impacto da Utilização de Mecanismos de Segurança em Nodos Sensores (WCSF 2004, Fortaleza, outubro de 2004)

CONSUMO DE ENERGIA PELO RÁDIO

Pacote	# bytes	Tempo (ms)	Energia (mJ)	aumento (%)
Padrão	36	15,00	1,215	-
Com autenticação	37	15,40	1,247	2.6
Com criptografia e autenticação	41	17,10	1,385	13.99

Obs.: broadcast de mensagens com payload de 29 bytes a cada 250 ms

Criptografia em RSSF: implementações e impactos nos nós sensores

IMPACTO NA CPU AO CIFRAR O CONTEÚDO DO PACOTE

Algoritmo	Tempo (ms)	Ciclos de clock	Energia (μJ)
SkipJack	2,16	15.925,2	51,84
RC5	1,50	11.059,2	36,00

IMPACTO NA CPU NO CÁLCULO DO MAC DO PACOTE

Algoritmo	Tempo (ms)	Ciclos de clock	Energia (μJ)
SkipJack	2,99	22.044,6	71,76
RC5	2,08	15.335,4	49,92

Criptografia em RSSF: implementações e impactos nos nós sensores

OCUPAÇÃO DE MEMÓRIA EM BYTES COM RC5

Modo Utilizado	ROM bytes	% ROM	RAM bytes	% RAM
Sem usar o TinySec	10510	8,018	444	10,840
Desabilitado	18206	13,890	918	22,412
Somente autenticação	18210	13,893	918	22.412
Criptografia e Autenticação	18212	13,895	918	22,412

OCUPAÇÃO DE MEMÓRIA EM BYTES COM SKIPJACK

Modo Utilizado	ROM bytes	% ROM	RAM bytes	% RAM
Sem usar o TinySec	10510	8,018	444	10,840
Desabilitado	17930	13,679	1174	28,662
Somente autenticação	17928	13,678	1174	28,662
Criptografia e Autenticação	17930	13,680	1174	28,662

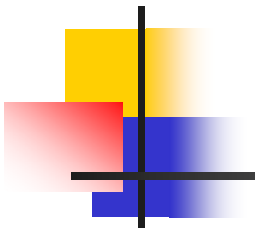
Criptografia em RSSF: implementações e impactos nos nós sensores

Impacto da Utilização de Mecanismos de Segurança em Nodos Sensores (WCSF 2004, Fortaleza, outubro de 2004) **- Conclusões:**

- Impacto pouco significativo em termos de consumo de energia devido ao processamento

- Impacto pequeno devido ao aumento no tamanho das mensagens

- Impacto aceitável no consumo de memória



Criptografia em RSSF: implementações e impactos nos nós sensores

Distribuição de chaves em RSSF

Criptografia em RSSF: implementações e impactos nos nós sensores

SPINS: Security Protocols for Sensor Networks (2002)

Motivação: impossibilidade de utilização de algoritmos de chave pública em nós sensores (memória e processamento)
Protocolos para redes de sensores: µTesla e SNEP

Utilização de chave mestra

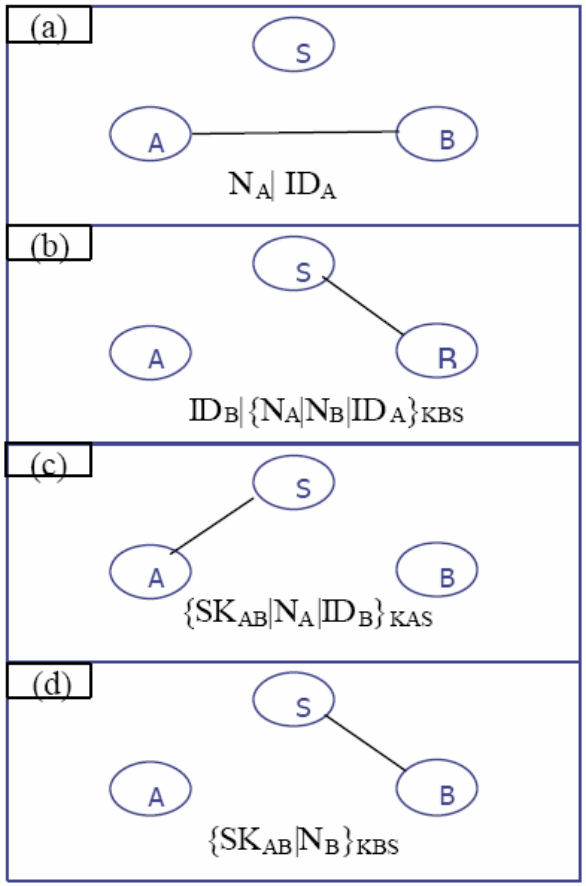
µTesla – broadcast com autenticação
Utiliza RC5 para implementar o MAC

SNEP – Secure Network Protocol (confidencialidade, autenticação, verificação de validade da chave)
Utiliza RC5 em modo CTR (algoritmo de bloco em modo fluxo)

Protocolos complexos (conceitualmente)

Criptografia em RSSF: implementações e impactos nos nós sensores

Distribuição de chaves com SNEP



Criptografia em RSSF: implementações e impactos nos nós sensores

Testes do SPIN com nós Smart Dust

Characteristics of prototype SmartDust nodes.

CPU	8-bit, 4 MHz
Storage	8 Kbytes instruction flash
	512 bytes RAM
	512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 Kbps
Operating system	TinyOS
OS code space	3500 bytes
Available code space	4500 bytes

Criptografia em RSSF: implementações e impactos nos nós sensores

Testes do SPIN com nós Smart Dust

Performance of security primitives in TinyOS.

Operation	Time in ms Fast implementation	Time in ms Small implementation
Encrypt (16 bytes)	1.10	1.69
MAC (16 bytes)	1.28	1.63
Key setup	3.92	3.92

RAM requirements of the security modules.

Module	RAM size (bytes)
RC5	80
TESLA	120
Encrypt/MAC	20

Criptografia em RSSF: implementações e impactos nos nós sensores

Consumo maior de energia é devido à transmissão, 98 % durante a transmissão de mensagens de 30 bytes:

Energy costs of adding security protocols to the sensor network. Most of the overhead arises from the transmission of extra data rather than from any computational costs.

71%	Data transmission
20%	MAC transmission
7%	Nonce transmission (for freshness)
2%	MAC and encryption computation

Criptografia em RSSF: implementações e impactos nos nós sensores

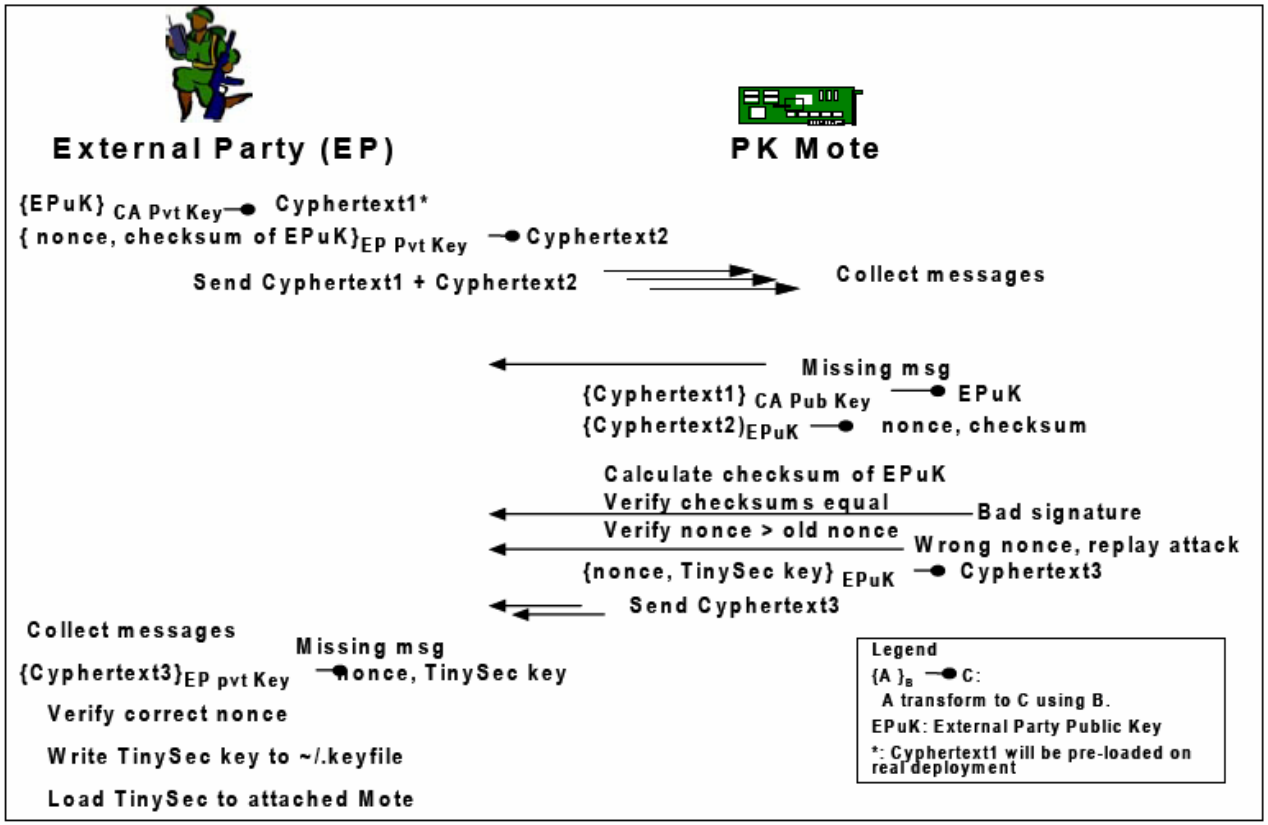
TinyPK: Securing Sensor Networks with Public Key Technology (2004)

Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, Washington DC, USA October, 2004

Infra-estrutura de chave pública para redes de sensores

Testes com RSA e Diffie-Hellman no MICA2 (TinyOS)

Criptografia em RSSF: implementações e impactos nos nós sensores



Criptografia em RSSF: implementações e impactos nos nós sensores

Resultados

Operação de cifração com chave pública no RSA (viável)

Foi utilizado e=3 (small exponent) para teste com RSA ($c = m^3 \bmod n$)

Table 1: RSA Small Exponent Operation Times

RSA Key Size	Time (sec)
512	3.8
768	8.0
1024	14.5

Operação de decifração com chave privada no RSA (chave ≥ 512 bits): dezenas de minutos (**INVIÁVEL** !)

$m = c^d \bmod n$ (d é muito grande)

Criptografia em RSSF: implementações e impactos nos nós sensores

Operação com chave pública no RSA: autenticação

utilizando certificados simplificados (operação equivalente à de um browser acessando um site seguro)

Operação com chave privada é inviável: não pode ser utilizada para confidencialidade ou distribuição de chaves

Criptografia em RSSF: implementações e impactos nos nós sensores

Testes com Diffie-Hellman (estabelecimento da chave secreta)

Table 2 Memory Utilization on Motes

	Modulus Size		
	512	768	1024
ROM (bytes)	12340	12376	12408
RAM (bytes)	847	1007	1167

Criptografia em RSSF: implementações e impactos nos nós sensores

Testes com Diffie-Hellman (estabelecimento da chave secreta)

$A = g^a \bmod n$ (troca de valores aleatórios)
$s = g^{ab} \bmod n$ (cálculo da chave)

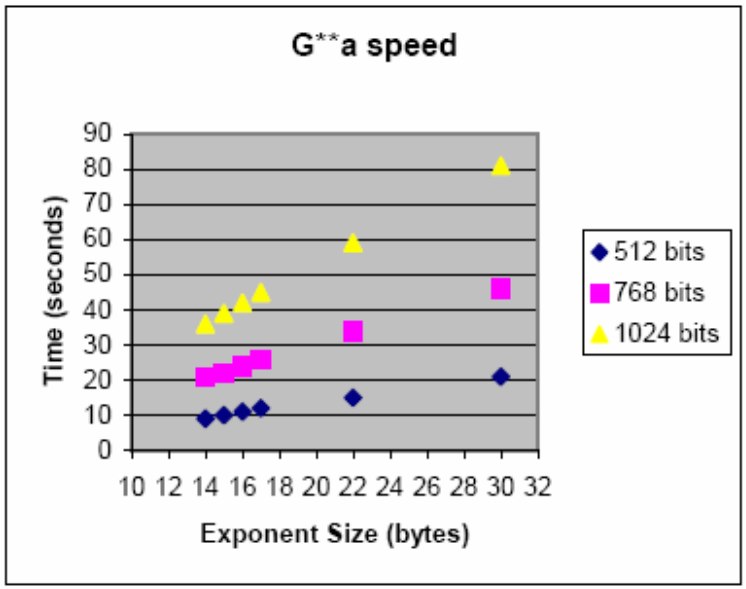


Figure 2 Execution time for first exponentiation

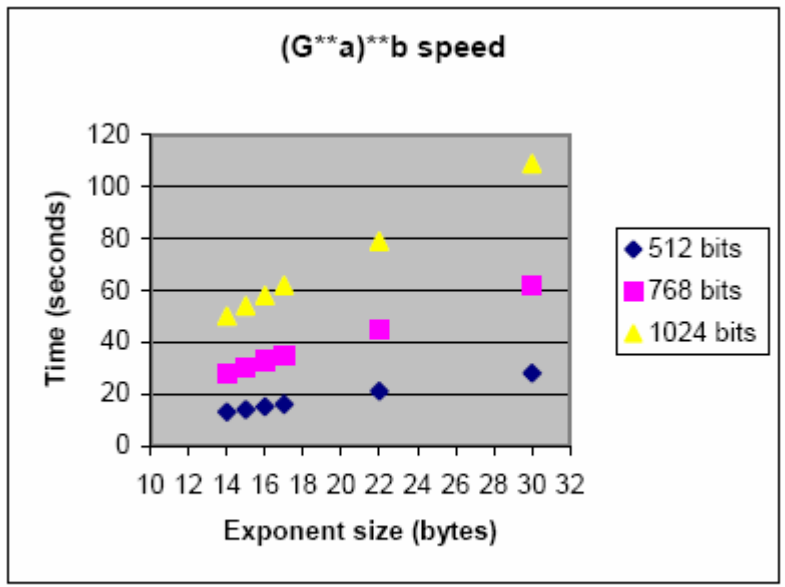


Figure 3 Execution time for second exponentiation

Criptografia em RSSF: implementações e impactos nos nós sensores

Testes com Diffie-Hellman (estabelecimento da chave secreta)

Os tempos são grandes mas o algoritmo de Diffie-Hellman é viável se a troca de chaves ocorrer com pouca frequência**.**

Podem ser utilizados módulo de 1024 bits e expoente de 160 bits para garantir uma razoável segurança (equivalente a 80 bits em algoritmos simétricos):

Bits of Security	Modulus	Exponent
80	1,024	160
112	2,048	224
128	3,072	256
192	7,680	384
256	15,360	512

Criptografia em RSSF: implementações e impactos nos nós sensores

A Public-Key Infrastruture for Key Distribution in TinyOS Based Elliptic Curve Cryptography (2004)

Proceedings of the First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, CA, October 2004

Implementação do Algoritmo de Diffie-Hellman com curvas elípticas (Mica2/TinyOS)

Observação dos autores: chave de Diffie-Hellman-ECC com chave de 163 bits oferece segurança suficiente.

Criptografia em RSSF: implementações e impactos nos nós sensores

General form:

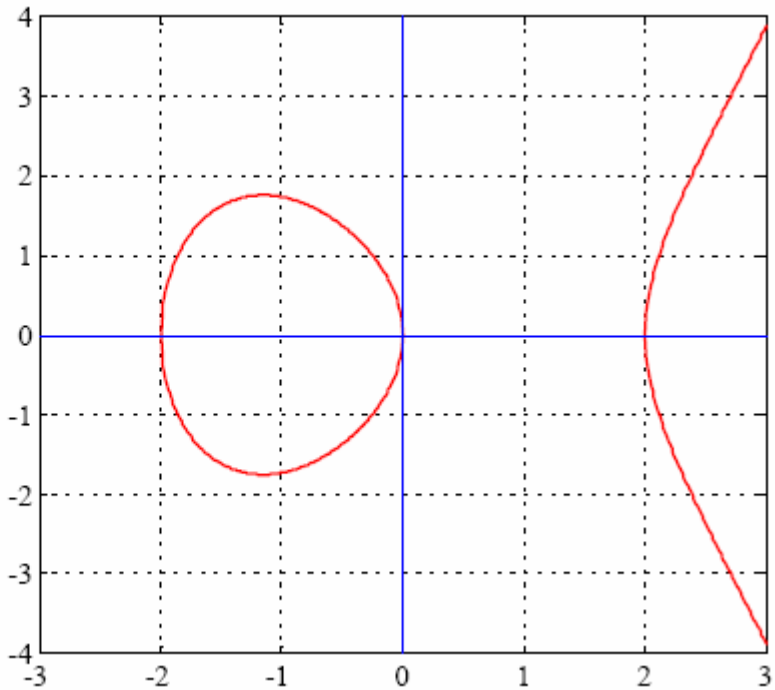
$$y^2 = x^3 + ax + b$$

Condition for distinct single roots:

$$4a^3 + 27b^2 \neq 0$$

Example:

$$y^2 = x^3 - 4x$$
$$= x(x-2)(x+2)$$

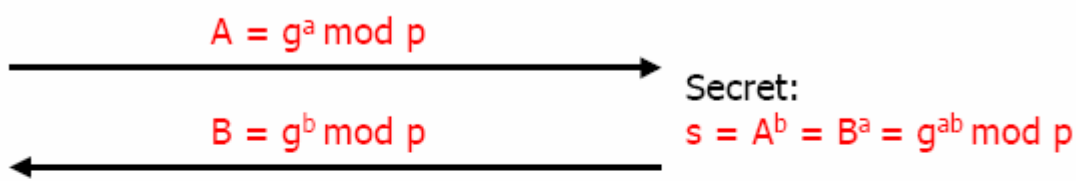


curvas elípticas sobre campos finitos (Galois Fields) GFp: pontos sobre a curva formam um grupo em GFp (valores inteiros entre 0 e p-1)

Criptografia em RSSF: implementações e impactos nos nós sensores

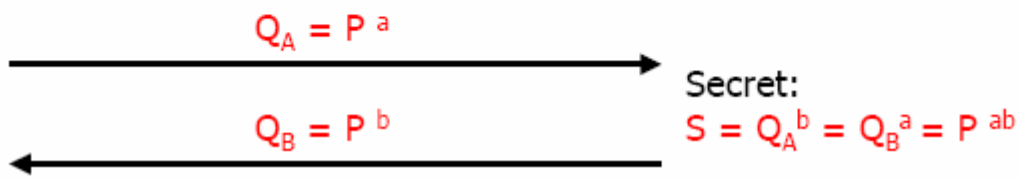
Diffie-Hellman-DLP

- Diffie-Hellman: Basis g and prime p



Diffie-Hellman-ECDLP

- Elliptic Curve Cryptosystem: ECC, basis point P and prime p



Criptografia em RSSF: implementações e impactos nos nós sensores

Desempenho da implementação de Diffie-Hellman-DLP

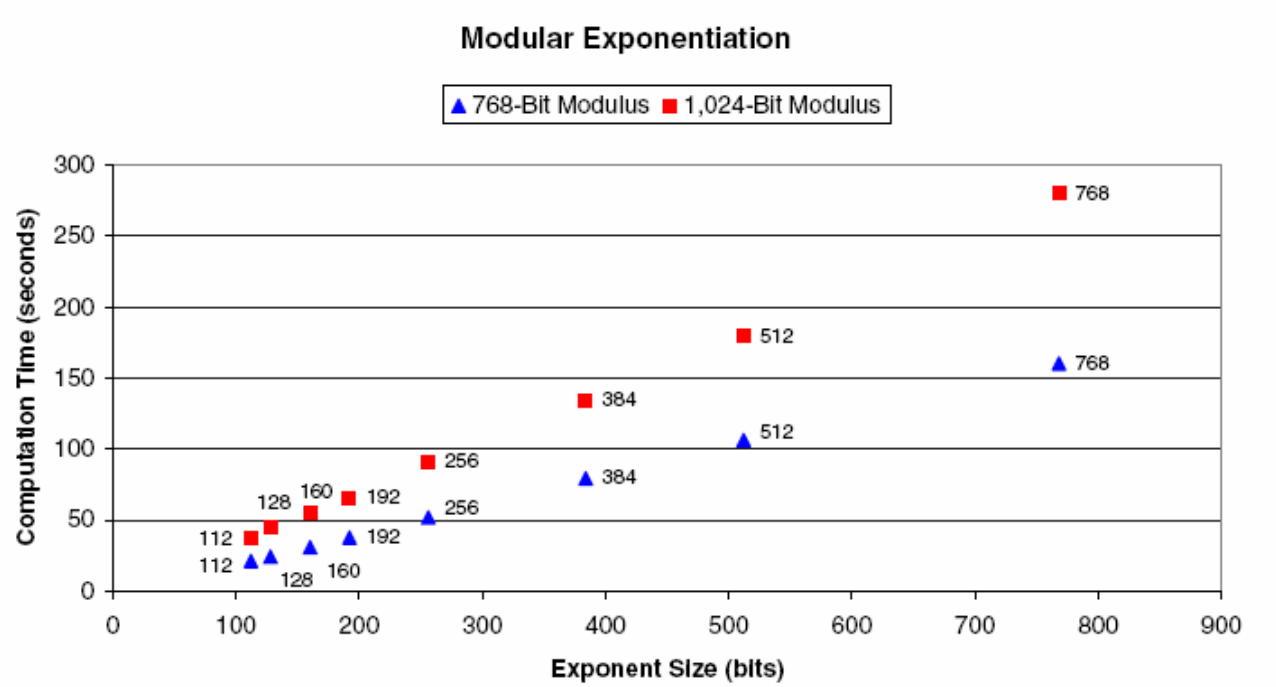


Fig. 3. Time required to compute $2^x \pmod{p}$, where p is prime, on the MICA2.

Criptografia em RSSF: implementações e impactos nos nós sensores

ENERGY CONSUMPTION OF MODULAR EXPONENTIATION, DETERMINED THROUGH INSTRUMENTATION OF AN IMPLEMENTATION OF DIFFIE-HELLMAN BASED ON DLP ON THE MICA2 WHICH COMPUTES 2^x (MOD p), WHERE x IS A 160-BIT INTEGER AND p IS A 1,024-BIT PRIME.

	1,024-Bit Modulus, 160-Bit Exponent
Total Time	54.1144 sec
Total CPU Utilization	3.9897×10^8 cycles
Total Energy	1.185 Joules

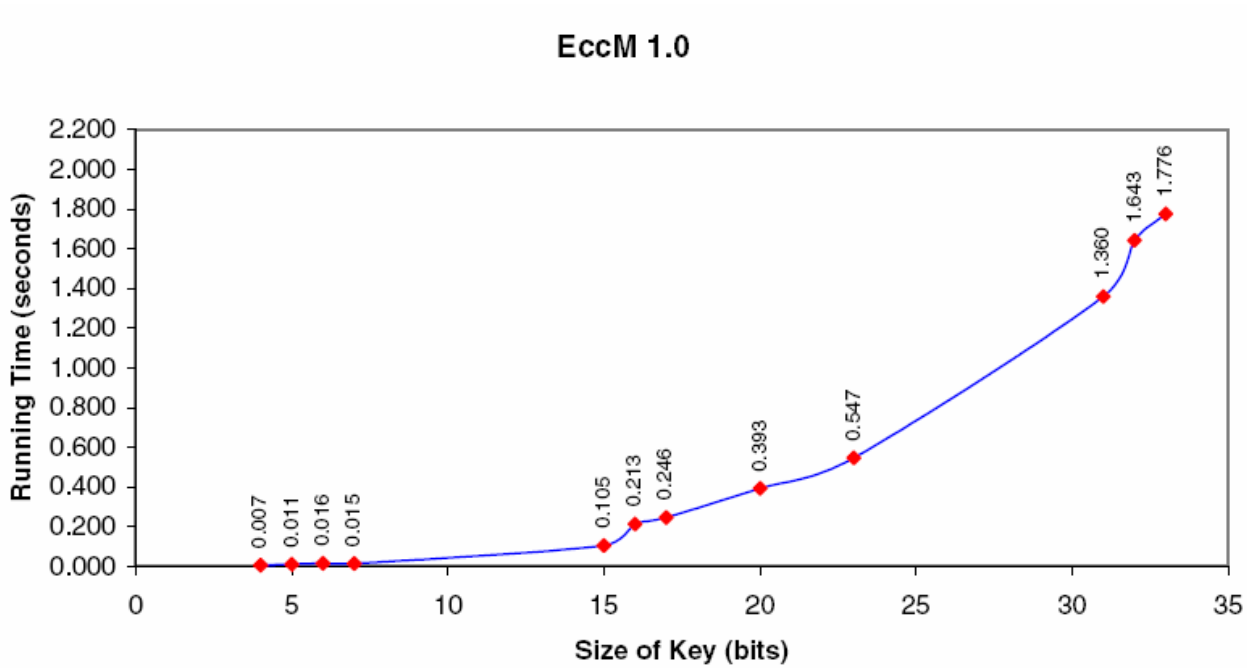
Criptografia em RSSF: implementações e impactos nos nós sensores

MEMORY OVERHEAD OF MODULAR EXPONENTIATION, DETERMINED THROUGH INSTRUMENTATION OF AN IMPLEMENTATION OF DIFFIE-HELLMAN BASED ON DLP ON THE MICA2 WHICH COMPUTES $2^x \pmod p$, WHERE x IS A 512-BIT INTEGER AND p IS PRIME. THE `.bss` AND `.data` SEGMENTS CONSUME SRAM WHILE THE `.text` SEGMENT CONSUMES ROM. STACK IS DEFINED HERE AS THE MAXIMUM OF THE APPLICATION'S STACK SIZE DURING EXECUTION.

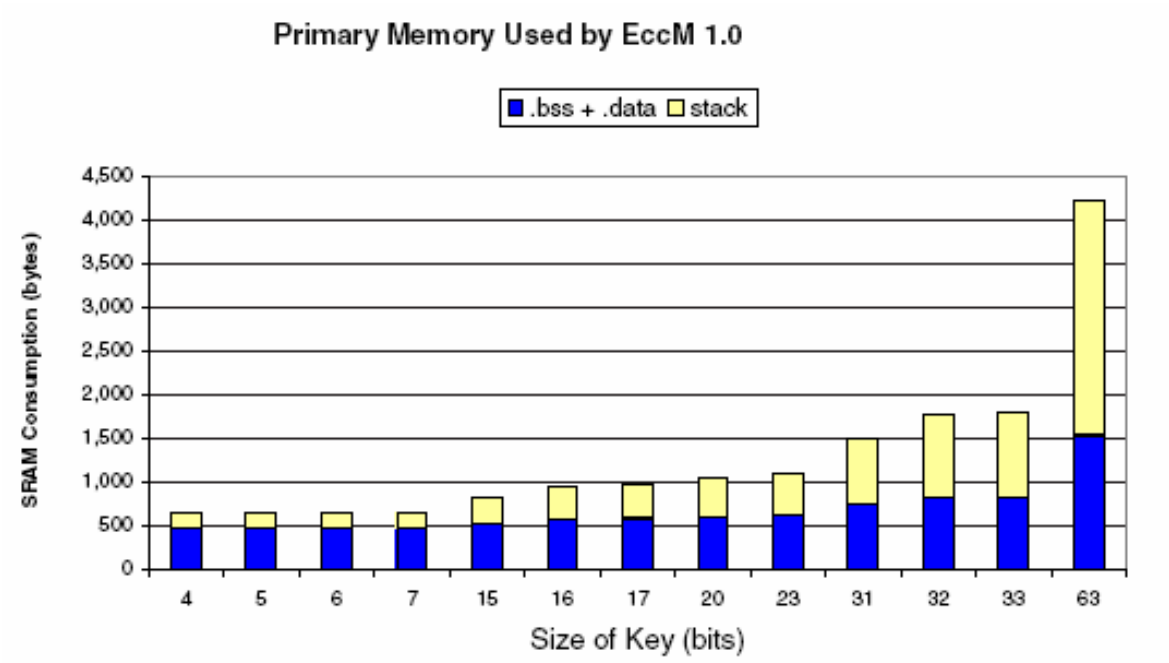
	768-Bit Modulus	1,024-Bit Modulus
`.bss`	852 B	980 B
`.data`	102 B	134 B
`.text`	11,334 B	11,350 B
stack	136 B	136 B

Criptografia em RSSF: implementações e impactos nos nós sensores

Desempenho da primeira implementação de Diffie-Hellman-ECDLP (acima de 63 bits ocorreu stack overflow)



Criptografia em RSSF: implementações e impactos nos nós sensores



Criptografia em RSSF: implementações e impactos nos nós sensores

Comparação das implementações de Diffie-Hellman-ECDLP (versão 1.0 e 2.0)

MEMORY USAGE OF ECCM 1.0 VERSUS ECCM 2.0. WITH ECCM 2.0, WE OBTAIN SIGNIFICANTLY MORE BITS OF SECURITY USING A REASONABLE FOOTPRINT IN MEMORY. THE .BSS AND .DATA SEGMENTS CONSUME SRAM WHILE THE .TEXT SEGMENT CONSUMES ROM. STACK IS DEFINED HERE AS THE MAXIMUM OF THE APPLICATION'S STACK SIZE DURING EXECUTION. MUCH OF THE INCREASE OF ROM'S CONSUMPTION IS THE RESULT OF ECCM 2.0'S ADDITIONAL FUNCTIONALITY.

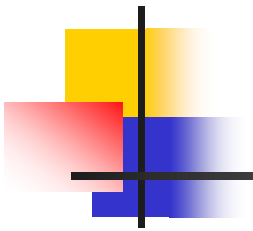
	EccM 1.0 (32-bit key)	EccM 2.0 (163-bit key)
.bss	826 B	1,055 B
.data	6 B	4 B
.text	17,544 B	34,342 B
stack	976 B	81 B

Criptografia em RSSF: implementações e impactos nos nós sensores

Desempenho de Diffie-Hellman-ECDLP (verão 2.0)

ENERGY CONSUMPTION OF ECCM 2.0, A TINYOS MODULE WHICH ALLOWS TWO NODES TO GENERATE PUBLIC AND PRIVATE KEYS (AND, THEREAFTER, TO USE THE SAME TO EXCHANGE A SHARED SECRET), DURING GENERATION OF A NODE'S PUBLIC AND PRIVATE KEYS.

	Private-Key Generation	Public-Key Generation
Total Time	0.229 sec	34.161 sec
Total CPU Utilization	1.690×10^6 cycles	2.512×10^8 cycles
Total Energy	0.00549 Joules	0.816 Joules



Criptografia em RSSF: implementações e impactos nos nós sensores

Conclusões

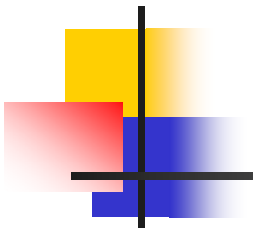
Criptografia em RSSF: implementações e impactos nos nós sensores

• A criptografia simétrica como mecanismo de confiabilidade, integridade e autenticação não causa impacto significativo.

• Criptografia de chave pública ainda sofre muita limitação devido ao hardware nos atuais nós sensores

• A implementação de PKI semelhante às utilizadas em redes tradicionais, que empregam algoritmos como o RSA (SSL, IPSec) ainda não é viável, é necessário simplificar as operações. Protocolos como os que utilizam Diffie-Hellman com chaves pequenas e RSA small exponente são viáveis. Desde que a troca de chaves não seja feita com freqüência.

• Não deve demorar muito para que sensores mais rápidos e com mais memória permitam a utilização de qualquer protocolo/algoritmo de redes tradicionais com segurança de alto nível. Desde que a troca de chaves não seja feita com freqüência para evitar consumo excessivo de energia.



Criptografia em RSSF: implementações e impactos nos nós sensores

Referências

Criptografia em RSSF: implementações e impactos nos nós sensores

Referências

Chris Karlof, Naveen Sastry, David Wagner; TinySec: A Link Layer Security Architecture for Wireless Sensor Networks; UC Berkeley; Proceedings of the 2nd international conference on Embedded networked sensor systems; Baltimore, MD; November 2004.

D. J. Malan, M. Welsh, M. D. Smith; A Public-Key Infraestructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography; Harvard University; *Proceedings of the First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON),* Santa Clara, CA, October 2004.

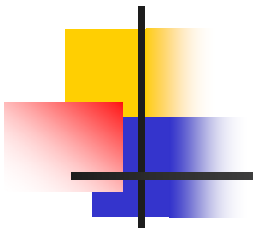
Criptografia em RSSF: implementações e impactos nos nós sensores

Referências

Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn1 and Peter Kruus; TinyPK: Securing Sensor Networks with Public Key Technology; BBN Technologies; *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, Washington DC, USA October, 2004*

ADRIAN PERRIG, ROBERT SZEWCZYK, J.D. TYGAR, VICTORWEN and DAVID E. CULLER; SPINS: Security Protocols for Sensor Networks; UC Berkeley; Wireless Networks, 2002.

Cláudia B.L.N. da Silva, Eduardo J.P Souto, Germano de F. Guimarães, Reinaldo C.M. Gomes;Djamel Sadok[1], Judith Kelner; Impacto da Utilização de Mecanismos de Segurança em Nodos Sensores; WCSF 2004, Fortaleza, outubro de 2004.



Criptografia em RSSF: implementações e impactos nos nós sensores

Perguntas ?