A Framework for Security on NoC Technologies

Catherine H. Gebotys,
Robert J.Gebotys*
Elec.&Comp. Engineering,University of Waterloo,
Psych, Wilfrid Laurier University*
cgebotys@optimal.vlsi.uwaterloo.ca

ABSTRACT

Multiple heterogeneous processor cores, memory cores and application specific IP cores integrated in a communication network, also known as Networks on chips (NoCs), will handle a large number of applications including security. Although NoCs offer more resistance to bus probing attacks, power/EM attacks and network snooping attacks are relevant. For the first time a framework for security on NoC at both the network level (or transport layer) and at the core level (or application layer) is proposed. At the network level, each IP core has a security wrapper and a key-keeper core is included in the NoC protecting encrypted private and public keys. Using this framework, unencrypted keys are prevented from leaving the cores and NoC. This is crucial to prevent untrusted software on or off the NoC from gaining access to keys. At the core level (application layer) the security framework is illustrated with software modification for resistance against power attacks with extremely low overheads in energy. With the emergence of secure IP cores in the market and nanometer technologies, a security framework for designing NoCs is crucial for supporting future wireless internet enabled devices.

1. Introduction and Related Work

As nanometer technologies become a reality, future wireless internet enabled devices will be increasingly powerful supporting many more applications including one of the most crucial, security. Design methodologies must be developed for secure software IP and hardware IP especially on portable devices where energy dissipation, and cost are driving design constraints. Furthermore as networks move to the chip level, increasing the complexity of the system on a chip (SoC), security increases in importance. Architectures must be designed which protect the user's private key from attackers, in particular untrusted software resident from invasive computing or untrusted external devices such as EM reading devices or hostile IP hosts. In general it will be of primary importance to additionally protect the secure IP cores from untrusted software and prevent exposure of keys outside of the SoC and even within the SoC communication network. Although there is new research studying networks on chips (NoC) [2,16], there is a lack of research studying how security would be integrated into these SoCs. Furthermore outside of smartcard research[5,13] (which typically is limited to cheaper 8-16 bit processors) few researchers have examined secure implementations of authentication software under the threat of power attacks on complex-parallel DSP processor cores. Design for security involves secure design of hardware cores, secure algorithm design, and secure network protocols, all important for NoCs of the future.

This paper illustrates a general security architecture for NoCs. and at the processor core level, design of security software with low overheads in performance, energy, yet resistant to power/EM attacks. At the network level, public key cryptography is implemented for communication between IP cores on the NoC which are involved in security applications. A security wrapper for each core and a central key-keeper core are proposed to ensure that unencrypted keys do not leave any core and the NoC itself. Security for NoCs has not previously been studied, however this approach has advantages for IP core vendors providing further protection of not only their hardware IP but also software which will run on their IP core. At the core level software design for resistance to power/EM attacks is illustrated, which unlike previous research has very low overheads in performance. This paper for the first time presents a security architecture for NoCs at both the network and core level which is crucial for future nanometer technologies.

NoC research has emerged as an important and growing area of interest. Researchers are developing tools to customize or reprogram the communication network design[2,16] as well as investigating reliability issues. Network layers for NoCs have been suggested in [2], specifically: the physical layer, architecture and control layer (data link, network and transport) , and software layer (application and system). NoCs in general will be highly constrained by reliability in addition to power dissipation and error correcting codes will be a necessity[2]. However security or security protocols have not been researched for NoCs.

Researchers have examined ASIC implementations of various encryption algorithms[19], system architectures for wireless security[9], and recently a number of secure IP cores have emerged such as AES cores, SHA-1, 3DES cores and others[17]. Initial studies of DSP processors for encryption applications have also indicated their usefulness[2,5,7]. Existing SoCs today have 4-16 DSP cores on them and NoCs are expected to have programmable cores such as numerous DSP cores as well as secure cores.

IP core protection includes both hardware protection such as watermarking techniques and simulation with software module IP protection in a CAD design environment such as [15]. In the later case public key cryptography is used to support simulation of chips which include software IP modules as well as hardware IP cores. A JavaCAD environment is described for designers which protects IP yet simulates designs which include hardware IP cores running third party IP software modules for example.

Power or electromagnetic (EM) attacks of NoCs may be releveant since cores may have separate power pins and EM radiation from the communication network may be significant. Researched power attacks of smart cards, have utilized general purpose processors with low clock frequencies [1,8,12,13,15] and more sophisticated processors with parallel instruction execution have not been reported in the literature. Researchers have suggested security against power attacks be achieved through 70% increase in computational cost[3]. However in portable devices, energy dissipation is very important hence a general purpose processor not optimized for power dissipation will not be suitable for running encryption in wireless communication.

Thus a framework for security is crucial for future NoC designs and a necessity for designing with secure IP cores available on the market today. This research presents a security design at the network level (or transport layer) and at the core level (or application layer). At the network level the security design utilizes a special key-keeper core and a network key on the NoC. This security methodology is independent of the type of communication network on the chip in general. The methodology ensures no unencrypted keys leave cores or the NoC and additionally supports IP secure cores running only trusted software. At the core level (or application layer) modification of security algorithms is illustrated to ensure their resistance to power/EM attacks unlike previous research with extremely low overheads in performance and energy. The next section will describe the security framework for NoCs at the communication network level, followed by security framework at the core level in section 3, specifically for software running on a DSP processor core.

2. Security for NoCs : Transport Layer Security

NoCs typically consist of a number of cores (memory, processor, DSP, ASIC, ASIP, etc) connected together by some (re)configurable network. The communication network may vary significantly from packet based communication to global bus communication[2]. The NoC will be required to support security for several reasons such as: 1) wireless communication or IP enabled applications requiring user authentication, encryption; 2) product authentication (preventing counterfeit products); 3) NoC chip authentication; 4) even IP-core authentication on the NoC itself.

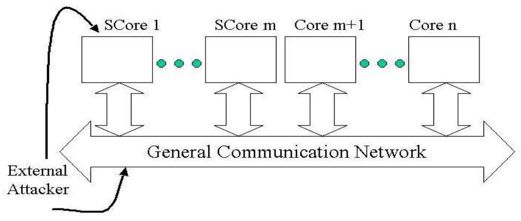

Figure 1. An example of a general NoC and security attacks.

We will assume NoCs have *m* secure cores (SCore in figure 1) and another *n-m* other cores, where *m≥1*. A secure core is defined as a hardware IP core which can execute one or more security applications (such as encryption, authentication, key exchange, etc) and it may also be able to execute other general non-security application as well.

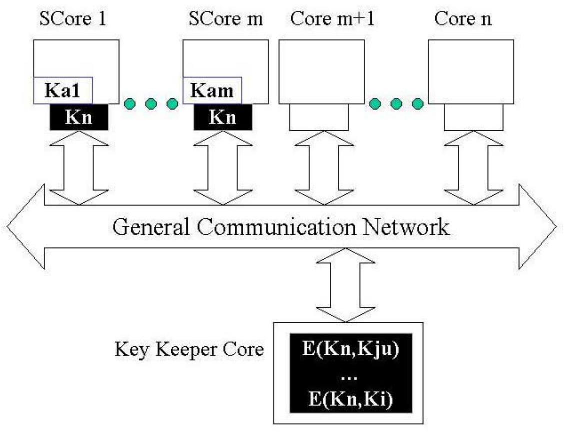

Figure 2. Security framework at network level protecting keys.

For *m=1*, the secure core would be solely responsible for authenticating messages or executables as well as (de)encryption and all other security applications. An advantage of this design is that once the users private key is transported to the core it can remain there and not require any further movement (unless new user keys are required). The possible disadvantages of this approach would be that it may be more vulnerable to power/EM attack, may be costly (since core may be quite large) and may degrade performance (since it must be queued for security applications). For example if a sole power pin was connected to this security core, power attacks may be possible, see top arrow in figure 1 (or detecting over which area of the chip the security processor is located to monitor for EM attacks). Secondly the secure core when not in use would be costly, taking up significant silicon area (equivalent to security chips currently on the market) and it may not be flexible for use in other applications (or possibly future security applications).

Alternatively for *n≥m≥2* security applications could run over any number m of secure cores in the NoC (depending upon the energy available, level of security needed, availability of cores). This later arrangement is also better for security throughput and resistance to power attacks, since typically the secure cores would have separate power pins and the attacker would not know at a time which secure core was executing the security application. This NoC scenario suits current secure IP cores on the market today as well where authentication and encryption requires several cores (AES, SHA-1, etc [17]). Additionally security applications could be assigned to secure cores randomly as they become available.

In general m secure cores, *m≥1*, and *n-m* cores can be supported in the proposed framework. The NoC should in either case be resistant to external attacks which can obtain data from the communication network (through external I/O pins attached to the communication network or EM radiation from it), shown as bottom arrow in figure 1.

Key distribution is important, even for *m=1* the single secure core, since keys are updated from time to time, the NoC will require a mechanism to allow downloading of the key onto the chip and into the memory of the secure IP core. The proposed security framework at the network level will be described next and supports key protection, and protected transmission on the communication network.

The network level framework for security is based on symmetric key cryptography, where each secure core (i) has its own security wrapper, along with a network key, Kn, stored in non-volatile memory within the security wrapper, see shaded boxes in figure 2. A key-keeper core in the NoC is required which is responsible for secure key distribution on the NoC. This core is also a secure core (and therefore has its own security wrapper). It keeps encrypted keys , see figure 2, such as encrypted keys for individual applications, encrypted user private keys (Kju) (ie. authentication or encryption keys to be used for decryption of audio, video, email, ebanking or signature generation or authentication of outgoing messages, etc) and other users public keys (Ki) (ie. for encryption of outgoing audio, video, email, signature verification, etc).

The network key (Kn) allows a IP core to receive application keys from the key keeper confidentially through the communication network on the NoC. Let E(a,b) represent encryption of b using key a, and D(c,d) decryption of d using key c, where both E(), D() are performed in the security wrapper. Consider performing an authentication application (such signing a message to be sent in an ebanking application) secure IP core x requires the private key, Kpu, of the user (or other user private keys, Kju). The user is the owner of the portable device containing the NoC. If the key-keeper core sends the unencrypted key directly over the communication network an external attacker snooping the bus (or another core in the NoC running untrusted software) may obtain it. The key transfer from the key-keeper core to secure core x on the NoC works as follows:

1. the secure core x sends a request for the users private key, Kpu to the key-keeper core.
2. To secure the transmission of this private key, the key-keeper core sends the encrypted key E(Kn, Kpu), from its memory, as a message (depending upon the network, it could be a packet for example) through the communication network. (Since only the secure cores have the security wrapper capable of decrypting the message, and they run only trusted software, the key transmission is safe.)
3. Secure core x receives the message and decrypts it (in the security wrapper) using the network key D(Kn , E(Kn,Kpu)) = Kpu to obtain the users private key, Kpu.

The decryption is performed in the IP core security wrapper, shown as shaded boxes in figure 2. Since this decryption is for the NoC network only, it can be very fast and utilize small area on the NoC, such as required by RC4 (de)encryption[11]. The key-keeper core stores only encrypted keys, such as the users private keys and public keys, including secure core public keys, however no key leaves the key-keeper core unencrypted.

New authenticated user private and public keys can be downloaded to the NoC in this security framework. The device with the embedded NoC would receive a new-key-message containing the encrypted new key, $K^{new}[p]u$ (which represents $K^{new}pu$ or $K^{new}u$). For example, the new-key-message $E_{aes}(K[p]u, K^{new}[p]u)$ could be encrypted by the users old public/private key, K[p]u, where $E_{aes}()$ is an encryption algorithm such as AES[4]. Updating the users private or public key so that it's stored in the key-keeper core would work as follows:

1. The key-keeper of the NoC would receive this new-key-message, $E_{aes}(K[p]u, K^{new}[p]u))$, and send it to a secure core x (which for example specializes in AES) for decryption
2. The key-keeper would then send the encrypted key, E(Kn,K[p]u), to the same secure core.
3. The secure core x would decrypt the encrypted key to get D(Kpn, E(Kn, K[p]u))=K[p]u in the security wrapper. Then computational units in the AES secure core would decrypt the new-key-message with this key, $D_a(K[p]u, E_{aes}(K[p]u, K^{new}[p]u)) = K^{new}[p]u$ using decryption algorithm AES.
4. The secure core would then send the result of the decryption (the new users key) back to the key-keeper by encrypting it in its security wrapper with the key-keepers public key, Kn. Specifically it sends $E(Kn, K^{new}[p]u)$.
5. The key-keeper core would receive $E(Kn, K^{new}[p]u)$ and store it

In this example authentication would also be required, to verify the users new private/public key pair did in fact come from valid authorities. This could be performed by computational units within the same secure core or a different secure core possibly in parallel (depending upon how the signature was formed). In all cases described authentication of messages sent from/to key-keeper to/from secure cores could also be performed in the security wrapper, assuming that authentication would not degrade performance of the network.

The symmetric key cryptography system could also support other uses. For example secure cores could receive new software upgrades from their IP core vendors as well. This is performed by sending to the NoC encrypted (and optionally authenticated) executables (encrypted with the private key of the secure core, Ka in figure 2 [and signed with the private key Ka for optional authentication which only the vendor of the IP core can do since they are the only ones who know the secure core's private key, Ka]). Again this upgrade can be performed remotely or wirelessly. Decryption or authentication (for example with crypto checksums[14] to ensure all security code executables are authenticated) could occur within the core (not the security wrapper) to safeguard the private core key. The core private key may also serve to identify the core, version, etc.

Additionally the vendor of the IP core can use the core's private key to prevent illegal use of the core in unauthorized NoCs (which have not paid license fees etc for use of the core). The core vendors would create an activation key (K_activation) for their core to be used in the NoC from the cores Ka and the IDs of the other cores in the network. Legal uses of the core would receive this activation key from the core vendor and store it encrypted in the key keeper core. For example on reset of the NoC the secure core would receive an activation key (K_activation) from the key keeper and verify this key by calculating its own security function

(F(), which would be encryption for an encryption core, etc) with its private key (Ka) and the concatenation of all other core IDs (obtained from polling all other cores) on the NoC. For example if K_activation = = F(Ka, core ID 1|| core ID 2 ||…core ID n), then the secure core will function properly else it will not operate and shut down permanently. Standards would have to exist so that each core would have a core ID number.

Ensuring confidentiality of communications on the on-chip network, and authenticity of executables which utilize private keys, a secure SoC environment for security applications is possible. Next the core level security will be addressed illustrating how software can be modified to further ensure security of private keys.

3. Security for Cores: Application Layer Security

Security within the secure IP cores is also important. The previous section provided a framework for ensuring that key's are safe within the communication network. This section will describe further security measures which could be taken within the core. An example of this methodology is illustrated on a DSP core running elliptic curve cryptography which provides strong yet efficient security for present and future technologies. Elliptic curve cryptography[6] can be used for encryption, key exchange and authentication (signature generation and verification). In these applications point multiplication is the key component which utilizes the users private key and thus requires additional security in its implementation. This section will examine the power trace of this component running on Star*core 140 DSP processor core[8] for NIST approved elliptic curve $y^2+xy=x^3+ax^2+b$ over 163 bit binary fields ($F_{2^{163}}$) using prime polynomial $x^{163}+x^7+x^6+x^3+1$ [10] (using affine coordinates).

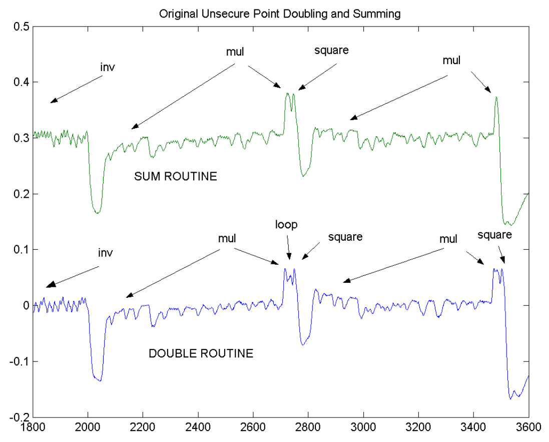

Figure 3. Power traces of partial point summing and point doubling algorithms in top and bottom graph respectively.

Elliptic curve point multiplication (or computation of xP where P and xP are both points on the elliptic curve and x is the secret key) is used in session key generation, authentication (signature generation, signature verification), and encryption. In elliptic curve cryptography, there is no multiplication operation, hence the only method for computing xP is by doubling and adding (ie. if $x=0101$ then $4P+P$ is computed in the algorithm, two doubles and one add).

In point multiplication, whether point doubling or doubling and summing is performed is key-dependent. Hence for a secure implementation it should not be detectable whether a point doubling or summing is being performed. Hence the software was designed utilizing redundant operations within the doubling and summing to make them look equivalent. Specifically 'equivalent' means that each cycle (after the field inversion since it is data-dependent), the processor performs the same instructions whether the application is point doubling or point summation (however the data will be different). Any final timing differences were resolved through redundant operation insertion. Power traces were obtained from running the original algorithm on the DSP core. Figure 3 shows the original unmodified point doubling and point summation routines running on SC140. The field multiplications (mul), field squaring (square), and important loops (loop) are identified. Figure 4 shows the modified software where point doubling and point summation were designed to be equivalent.

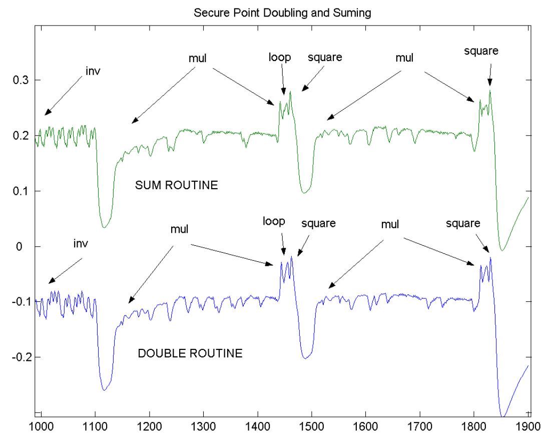

Figure 4. Power traces of modified secure point summing (top) and doubling (bottom) algorithm.

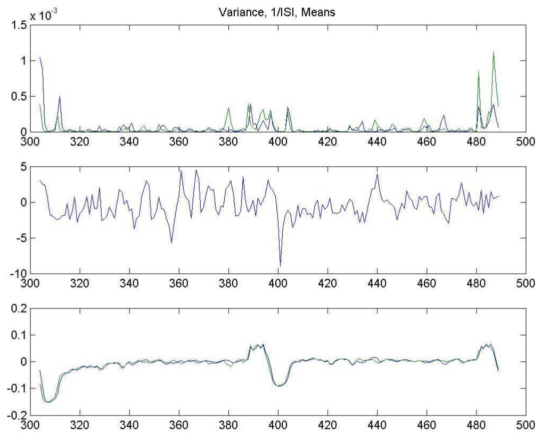

Figure 5. Variances, 1/ISI metric, means of double and sum power traces.

The routines have similar power traces, however matlab analysis was performed to confirm this. The mean of point doubling power subtraces was subtracted from the mean of point summing power subtraces and divided by the error in the difference of means when the two sets come from the same distribution[18]. Any peaks in this value would indicate a significant difference between the point double and point summation power traces. This approach indicated that the two sets of traces were very similar and no notable differences occurred as shown in figure 5, where two means are shown at bottom, middle shows difference of means and top shows variance of sets. Apart from the drop in at 400 (which occurs due to very low variances at this point), the variation ranges from +5 to –5 which indicates good agreement[18]. Using these software techniques for improving the implementation security provides a solution with lower overheads in performance and energy dissipation. Previous research has overheads of 70% in performance[3], whereas our overheads for the binary fields are 1% in performance and 1% energy dissipation and 11% in code size.

4. Discussions and Conclusions

This paper presented for the first time a framework for security on NoCs. It presented a symmetric-key based cryptography design for securing the communication network within the NoC. Authentication, encryption, key exchange, new user keys, public key storage, etc are supported in this framework. Additionally there are benefits to the IP core vendor for this setup including more protection of software running on their hardware IP core (a solution to untrusted or invasive software). Another advantage of this new framework is that copying IP cores without the activation keys (provided by vendors) will prevent their illegal use in SoCs. The symmetric key system simplifies the NoC design process and key management for security in SoCs. It also helps to protect hardware IP (ie. preventing unauthorized access to it) as well as software IP (ie. preventing modification of software IP through authentication). If user keys were not expected to change, each secure core could also have copies of these keys in local non-shared nonvolatile memory (however this situation is very restrictive and costly, and future NoCs should support new user keys). Key management is still necessary since public keys would still be distributed and sent to secure cores.

The security wrapper could be standardized and implemented by the NoC designer or core vendor. Specific encryption and authentication algorithms for the security wrapper have not been described, however known low complexity and fast algorithms do exist such as RC4, and different security wrappers could be designed based upon the NoC requirements (where all secure cores in the NoC would have the same secure wrapper). Since reliability is crucial in NoCs of the future its possible that combined decryption and error correcting would be highly advantageous[20]. The security framework could also be extended to support dynamic nature of NoCs where depending upon current battery energy levels and QoS of messages, secure cores could run at different clock frequencies, power levels, etc and operate different encryption algorithms (varying from weak to strong security). Additionally during reset, a new NoC key could also be generated using a pseudo random number generator with a key exchange algorithm to ensure further security.

This study presents for the first time a framework for security of NoCs by providing network level symmetric-key cryptography for key distribution and at the core level by illustrating modification of software with extremely low overheads for added security against power attacks. Unlike previous research, which has not studied security on NoCs, this scheme has added advantages for protection of software and hardware IP. For core-level security, software design is illustrated and verified with real power traces for possible resistance to power attacks, with <11% overheads unlike previous research which suggested 70% overheads in performance. This research is crucial for supporting a methodology for designing security for NOCs which will be prevalent in wireless IP-enabled devices designed with nanometer technologies of the future. This research was funded in part by grants from NSERC, and Motorola.

REFERENCES

[1]P.Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", LNCS, 1998.

[2] L.Benini, G.DeMicheli, "Networks on Chips: A New SOC Paradigm" Computer, Vol.35, No.1, Jan 2002, pp.70-78.

[3]P.Liardet, N.Smart "Preventing SPA/DPA in ECC systems using the Jacobi Form", LNCS 2162, May 2001, pp391-401.

[4] AES reference http://csrc.nist.gov/encryption/aes/rijndael

[5]O.Kommerling, M.Kuhn "Design principles for Tampter-resistant Smartcard Processors", USENIX workshop on Smartcard Technology, 1999.

[6]D.Hankerson, J.Hernandez, A.Menezes "Software Implementation of Elliptic Curve Cryptography over Binary Fields", White Paper, www.certicom.com, 2000

[7] S.Dusse,B.Kaliski, "A cryptographic library for the Motorola DSP56000", vol 473, LNCS, May 1990, pp.230-244.

[8] "Star*Core 140 DSP Core Reference Manual", Motorola/Lucent, Sept 1999.

[9] S.Ravi, A.Raghunathan, N.Potlapally "Securing Wireless Data: System architecture challenges", ISSS, 2002, pp.195-200.

[10] IEEE Std 1363-2000, IEEE Standard specifications for public-key cryptography, IEEE computer SoC. 2000.

[11] A.Stubblefield, etal. "using the fluhrer, mantin and shamir attack to break WEP", AT&T Lab Tech Rept TD-4ZCPZZ, 2001.

[12] P.Kocher, J.Jaffe, B.Jun "Differential Power Analysis" Crypto'99, LNCS 1666, 1999.

[13] T.Messerges, E.Dabbish, R.Sloan "Investigations of Power analysis attacks on Smartcards" USENIX workshop on Smartcard Technology, 1999.

[14] B.Yee "Using Secure Coprocessors" PhD thesis, CMU-CS-94-149, 1994

[15] M.Dalpasso, A.Bogliolo, L.Benini "Hardware/Software IP Protection", Proceedings of DAC, 2000.

[16] P.Guerrier, A.Grenier "A Generic Architecture for on-chip packet-switched Interconnections" Proceedings of DATE, 2000.

[17] Data Security Silicon IP cores, www.heliontech.com

[18] C.Gebotys, R.Gebotys "Secure Elliptic Curve Implementaions: An analysis of resistance to power-attacks in a DSP processor" Proceedings of CHES 2002, pp.114-128.

[19] H.Kuo, I.Verbauwhede "architectural optimization for 1.82Gbps VLSI implementation of the AES Rijndael algorithm" CHES 2001, LNCS 2162, May 2001, pp.51-64.

[20] C.Lam, G.Gong, S.Vanstone "message authentication codes with error correcting capabilities", http: //cacr.uwaterloo.ca, 2002