T-IBE-T: Identity-Based Encryption for Inter-Tile Communication

Alexander Würstlein, Wolfgang Schröder-Preikschat
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

ABSTRACT

T-IBE-T applies identity-based encryption (IBE) to inter-tile communication in tiled multi-processor system-on-chip (MPSoC) hardware architectures. There, a network-on-chip (NoC) enables communication among topologically disconnected parts of an application. When such an MPSoC is shared while tiles are exclusively allocated, with malicious applications present, the NoC becomes a weak point. Yet minimal memory footprint, as well as scalability, are necessary preconditions on any security mechanism in this scenario. We show that T-IBE-T provides each component such as the tiles' OS instances, applications and tile application instances with a secure key exchange while ensuring asynchronicity, minimal latency and providing no-cost key distribution.

ACM Reference Format:
Alexander Würstlein, Wolfgang Schröder-Preikschat. 2019. T-IBE-T: Identity-Based Encryption for Inter-Tile Communication. In *12th European Workshop on Systems Security (EuroSec '19), March 25–28, 2019, Dresden, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301417.3312500

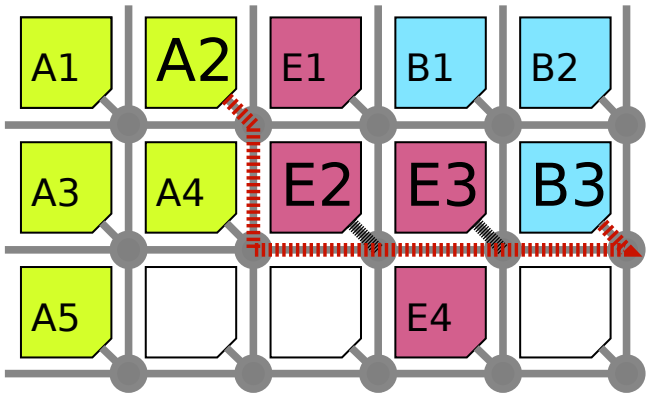

Figure 1: Tiled MPSoC system with running tasks from applications A, B and E. NoC communication (red, dotted) from task A2 to B3 needs to pass malicious tasks E2 and E3, where the malicious application E can eavesdrop and create frames.

1 INTRODUCTION

This work will describe T-IBE-T, an encryption system for the inter-tile communication in tile-based computing architectures utilizing identity-based encryption (IBE) to enable asynchronous low-latency low-overhead secure network-on-chip (NoC) communication. Tiled architectures are a growing trend in multiprocessor designs, combining aspects of traditional multiprocessors and computer networks [17]. NoC interconnect performance, latency and communication patterns are of utmost importance, lest they become a limiting factor to overall system performance. With the advent of core and tile counts in the thousands, old solutions like per-core copies of data structures also become resource usage problems that need to be solved anew. T-IBE-T provides a security mechanism sufficient to counter present threats yet with minimal impact on NoC performance and resource usage.

1.1 Tiled Architectures and common MPSoCs

Tiled architectures envision chips with an order of thousands of cores, grouped into tiles which comprise cores of identical makeup with locally shared resources. The communication between different tiles is mediated by a NoC that connects all tiles in a grid topology as shown in figure 1. Each application may be assigned resources which might be topologically connected and closed (meaning that all communication paths between tiles of one application do not pass tiles of another application), open or disjoint, such that at least some private communication of the application will pass hostile tiles belonging to an untrusted application. Renting out shared compute resources or sharing a multipurpose system among applications of varying provenance becomes more and more common, so attacks from untrusted applications have to be considered. Each tile is exclusively assigned to only one application at a time and assumed to be isolated such that the only means of communication and interaction have to be mediated by the NoC. This means that the attack scenario will be concerned with the malicious applications acting on the tile's NoC interface by reading and creating messages.

In this way, a tiled architecture is similar and representative of many non-trivial MPSoC architectures. With the growing complexity of such systems, traditional networked communication problems such as untrustworthy nodes on a network have to be considered in this new but similar domain. However, there are relevant differences between an MPSoC NoC and a traditional computer network, namely that there is a bootup phase in MPSoC NoCs, where we assume that the system setup procedure still has full control over all of the network and is able to perform initialisation routines necessary for later secure communication. Such a common boot and setup phase is not available in networks in general.

EuroSec '19, March 25–28, 2019, Dresden, Germany

ACM ISBN 978-1-4503-6274-0/19/03…$15.00
https://doi.org/10.1145/3301417.3312500

Furthermore, our tiled target architecture enables micro-parallelism by providing very lightweight tasks that execute a given code with associated data in a run-to-completion fashion on a remote tile. Thus, the very frequent spawning of new tasks on other tiles necessitates low-overhead initiation of communication sessions since participants may be rather short-lived. Therefore a major focus of this paper is to minimize per-connection, per-message and first-connection overheads. As an example, one of the most common patterns in such computation calculates the iterations over a loop in a number of tasks, after which the results are collected by one-off messages back to the creator. In the most extreme cases this means that interaction and synchronicity between participants of a communication are not desirable since the sender of a message may no longer exist upon reception, or the receiver of a message may not yet exist upon sending the message.

Thus an ideal security mechanism for such a system will allow asynchronous, into-the-future, posted and one-off communications with minimal interaction with third parties and minimal preparation. We claim that an IBE implementation such as T-IBE-T can offer these properties, better than more conventional alternatives.

1.2 Contributions

We claim the following contributions:

(1) T-IBE-T is the first practical application of IBE focusing on tiled MPSoCs
(2) a description of a practical IBE system setup process
(3) an analytical evaluation of IBE in contrast to other common encryption systems

1.3 Structure of this Paper

In section 2, a description of common symmetric and asymmetric cryptosystems will be followed by a short introduction into IBE, after which T-IBE-T is explained. Section 3 will evaluate T-IBE-T under an attack scenario and compare T-IBE-T's properties to state-of-the-art symmetric and certificate-based encryption systems. The final sections 4 and 5 will provide the reader with an overview of related works, our plans for the future, and ideas as well as a summary.

2 METHODS

This section will discuss conventional approaches to securing communications, then introducing IBE and the T-IBE-T approach.

2.1 Symmetric Cryptography

In the symmetric scenario, the preparatory phase prior to communication is used to create a number of symmetric encryption keys: Each pair of communication partners needs to secure a communication channel meaning that for each combination among partners $\mathbb{P}$ a key needs to be generated.

This, of course, will become untenable quickly, since the number of necessary keys to be pregenerated is $|\mathbb{P}|^2$. Pre-generation of a quadratic number of keys may be avoided if one chooses to just pregenerate a single key for each partner and a central authority which creates and distributes connection keys later on, akin to Kerberos. However, this central authority merely represents a shifting of preparation cost to runtime as well as a single point of contention at runtime. Each participant of a communication will have to contact the central authority to obtain the mutual connection key. Therefore these symmetric systems are limited in their scalability and usefulness, as section 3 will elaborate.

2.2 Asymmetric and Certificate-Based Cryptography

The certificate-based system employs a central certification authority (CA), which is used to sign the verified identity certificate of each participant. Therefore the preparation phase consists of the creation of a common trusted CA certificate that is made known to all participants, and the creation of a CA-signed identity certificate for each participant.

To communicate, a key exchange method such as the Diffie-Hellman (DH) key exchange is used to create a connection key upon establishing a connection. Such a DH exchange is secured by signing the respective packets sent in the exchange with the respective sender's identity certificate, thereby excluding the possibility of man-in-the-middle attacks on the key exchange. In this internet-type scenario, communication only needs to happen between the intended communication partners, the CA or any other party is not involved. The participants also need not know each other's certificate before the exchange, certificates may be sent along as a part of the exchange and are then trusted depending on the validity of the CA signature. However, for an initial DH exchange to succeed, at least one answer to the initial packet is necessary. For a sender-initiated message, this means that only the third packet in an exchange can transmit useful data to the receiver, unacceptable for our use case.

At the expense of forward secrecy, a key exchange may be simplified to what is known as the RSA key exchange. Knowing the recipient's RSA key, the sender randomly creates a symmetric session key and encrypts and sends this session key to the recipient, ideally along with the message encrypted using the symmetric key. This mode is alike the one used in email encryption systems like S/MIME or PGP. Importantly, meaningful data can be transmitted in a posted manner, without any necessary round trip even on the first contact between two communication partners. However, since the recipients public key has to be known, a key directory is necessary in some form, either as a central instance to be contacted before a connection or as a globally known key table. So the reduced number of network round-trips before a payload is an improvement at the expense of higher memory consumption.

2.3 Identity-Based Encryption

Identity-based encryption (IBE) was first hinted at by Shamir as an asymmetric encryption system where the public key of each participant would be a commonly known characteristic identification string ID [13]. The canonical example for such an ID would be the participant's unique email address. To contact Bob, Alice would need Bob's email address anyways, so for Alice to send an encrypted email to Bob, no additional knowledge of Bob's public key needs to be obtained since the key is just the email address. This obviates the need for key directories in posted, non-interactive communications.

Boneh and Franklin first introduced [2] a viable cryptosystem based on pairings that could be used to implement IBE, and numerous other implementations have been proposed since [1, 4, 15]. For this work, detailed knowledge of any IBE implementation is not required; therefore a schematic overview of a general IBE system will suffice. Since this work only makes use of general properties common to IBE systems, the ideas presented are applicable to all IBE implementations.

An IBE system will be initialized by choosing a public set of parameters *param* and a private master key M which should be known only to the public key generation service (PKGS), a trusted central service later placed on a separate operating-system-owned tile. Alice will then pick her ID_A and obtain a corresponding secret key d_A from the PKGS. To encrypt a message to Alice, Bob is assumed to know Alice's ID_A as well as the globally published *param*.

From a practical point of view, there are several noteworthy circumstances to be considered: Alice's private key will not only be known to Alice but also to the PKGS, meaning key escrow is a core property of IBE. However, there are IBE systems that lack this property [3], and in some use cases such as ours, key escrow is desirable, as will be discussed further down. Second, the *param* set has to be made known publically in such a way, that all participants use the same *param*. Otherwise, Mallory can provide Bob with a separate set $param^M$, impersonate the PKGS by generating a malicious master secret M^M and provide a malicious private key d_B^M to Bob, enabling Mallory to be man-in-the-middle to all of Bob's encrypted communications. Third, and most importantly, a scheme has to be implemented by which the PKGS can verify the identity of all private key requests and transmit private keys securely back to the requester. Consequently, a practical implementation of an IBE scheme will need to be accompanied by a signature scheme to ensure the integrity and authenticity of such requests.

2.4 Identity-Based Signatures

By Shamir, an identity-based signature scheme was described [13], implemented as what is now known and widely used as a certificate-based PKI system. By means of an asymmetric signature algorithm, a trusted certificate authority (CA) signs with its private key an identity certificate consisting of Alice's public key and identification string. A signed message to Bob will be accompanied by Alice's identity certificate, such that Bob can verify message being from Alice: The identity certificate has to be validly signed, verifiable with the CA's public key. The identity string has to match Alice's and the message signature has to be verifiable with Alice's public key from the identity certificate. Thus the only prior knowledge required of Bob is the CA's public key as a global system parameter and Alice's *ID*.

Signatures proving the identity of the sender and the integrity of the message have been known for a long time [13], are commonly used and suitable for the purpose of this work. This work will concentrate on the challenge of encryption to a recipient. The existence and application of sufficiently secure signatures on messages used in T-IBE-T is therefore implicitly assumed in the following.

2.5 T-IBE-T

In T-IBE-T, the system will pass through the following phases:

(1) *Setup*: Bootup, application initialisation, application and service key and *param* distribution
(2) *Start*: Resource acquisition, task key distribution, and task initialisation and start
(3) *Work*: Task operation and communication
(4) *End*: Task completion and resource reclamation

The setup phase will happen only once for the whole system and will be synchronized among all tiles. The following three steps will be performed for each task on each tile independently. Two things are important to note: First, it is assumed that each task has run-to-completion semantics, meaning that neither preemption nor preemptive multitasking will take place on a given tile. And second, the three task lifecycle steps may be repeated, meaning that after the end of a task, another task, possibly of another application, may be assigned the newly-freed tile and utilize it until that new task has ended, and so forth. Each task may create other tasks during it's lifetime.

Applications in this model start with bootup and end at the completion of their last remaining task. An application represents the set of tasks intended to fulfill a common purpose, the tasks of an application are usually mutually more trusted. Different applications and their tasks are seen as potentially hostile. However, the communication with different applications should remain possible if necessary.

The operating system (OS) of such a tiled computer system will function in a distributed manner as well. Each tile will have an OS instance that handles local system calls for the tile's current task, remote and global OS issues can be handled as well through a remote system call mechanism via NoC messages. Resource allocations like tiles, memory and I/O resources will be mediated and assigned by global OS mechanisms, also preventing and stopping malicious overuse.

For this work, the aforementioned mode of operation and its terminology simplify following descriptions. However, to utilize T-IBE-T in other contexts, a sufficiently strong concept of task and application isolation is the only prerequisite. Therefore we claim the applicability of T-IBE-T for other MPSoC systems and further similar scenarios.

The following parts will explain the necessary lifecycle stages, from the setup and boot process to the creation of a task which then can communicate with another task.

2.5.1 Setup and Boot. One major difference when compared to computer networks is that networks-on-chip will go through two global initialization phases: In the setup phase, a system image will be prepared that contains the operating system and possibly applications to be run as well as necessary data. The system image is then provided, e.g. via permanent storage, network or debug interfaces to the SoC for the boot phase. During this boot phase, the operating system retains complete control over the hardware, especially also the NoC. This means that for the boot phase before any possibly malicious application code could be run, the NoC is considered trustworthy since no attacker can yet be present on the NoC.

Table 1: Examples of *ID*s for a tile's OS instance, an application on that tile, and a task thereof; as well as a global OS service or global application *ID*

destination	`int8_t[]` *ID*	description
tile (17,3) (OS)	`'O',17,3`	<type>, <tX>, <tY>
... application B	`'A',17,3,'B'`	...<appNum>
... task (*i*let) 5	`'I',17,3,'B',5`	...<taskNum>
global OS service	`'S','PKGS'`	<type>, <service>
whole application A	`'W','A'`	<type>, <appNum>

In the setup phase, common static parameters *param* of the cryptosystem will be generated and made part of the system image. During the boot phase, those common parameters will then be used to create the master secret M and the dynamic parts of *param* as well as private keys for the system services, especially the private key generation service (PKGS). Each tile's operating system (OS) instance will be set up with *param* and an identity key for the tile's OS instance to use in remote system calls. Memory protection will be employed for this data to prevent manipulation or disclosure to the application. An application originates from a single task which can then create an arbitrary number of subsequent tasks on any tiles of the application's (variable) resource allocation. When an application A is initially created by the OS, a secret key for the application ID_A is created and placed into the initial task's address space for the application to use.

2.5.2 Task Start. When a tile's OS is sent a new task to execute, the task's code and associated data will be set up in a new tile-local unprivileged address space. The tile's OS will also obtain the secret identity key for the task in creation from the PKGS, placing the key into the task's address space, followed by execution of the task.

2.5.3 Work and Communication. After all setup steps described before, a task is able to encrypt data to another task by using the appropriate *ID* as the encryption key. A symmetric message key is randomly created and IBE-encrypted to the recipient's *ID* along with the symmetrically encrypted message. Replies to the original message use the same procedure with a different randomly created symmetric key, intentionally not creating dependencies between the different messages. This avoids creating synchronous behaviour in places where the application itself would be asynchronous. The necessary *ID* for the communication partner should be easy to infer from the knowledge that the sending partner already has, as described in the following section.

2.5.4 ID Naming Scheme. Communication is based on each participants *ID*, in routing as well as in cryptography, a correspondence that is the basic idea of IBE. Therefore an efficient and straightforward conversion from the routing, resource allocation, and task information to IBE *ID*s is necessary.

A tile is addressed by it's number, which can be split into coordinates within the tile grid, e.g. $(17, 3)$ would be the tile in column 17 row 3. The most straightforward representation here would be to use the same `int16_t tile` that is used in the routing hardware. All *ID*s in T-IBE-T are prefixed with an *ID* type marker that determines their size and makeup. Table 1 contains the envisioned types and their respective formats. This does not preclude later changes: All *ID*s in T-IBE-T are ephemeral in that they only need to be valid from system boot to shutdown.

2.5.5 Key Revocation. In cases of private key disclosure on an MPSoC system, an immediate shutdown may seem the safest option. Yet scenarios where availability is more important than security demand the possibility of key revocation. Often in IBE systems, a maximum validity time is made part of the *ID*. This approach has problems: First, all *ID*s and keys will need to be renewed at the end of their validity, leading to performance impact. Second, and more importantly, this maximum validity together with the PKGS's refusal to renew is not revocation in that the keys still remain usable until the end of the assigned period. In our MPSoC scenario, however, there is a simple and obvious solution: A signed NoC broadcast can be used to make the revocation of a key known to all participants. To enable the creation of a new key for a revoked *ID*, all *ID*s would then have to include a version number. The newly valid *ID* version would also be announced in the revocation broadcast.

3 EVALUATION

The evaluation scenario for the aforementioned approaches will be explained in this section.

3.1 Attacker Model

A malicious application task may, by abusing it's access to a tile's NoC interface, read network frames passing the interface, as well as create arbitrary network frames. However, we will not consider NoC flooding and denial-of-service type attacks since in our target architecture these attacks can be mitigated through NoC channel reservations. Changing NoC routing is out of scope since routing decisions are static hardware mechanisms of the architecture. Moreover, changing frame contents on-the-fly is prevented by flow control mechanisms of the network interface.

The OS instance on each tile is considered non-malicious, sufficiently separated, and privileged over the tile's malicious task. However, tile-to-tile communication between two OS instances or OS instances and applications can be attacked in the same way as application-to-application frames.

Considering this attacker model, T-IBE-T can be considered secure against the disclosure of message contents, if the chosen IBE implementation together with the symmetric cipher mode have this property. In a similar manner, the creation of arbitrary network frames by the attacker does not lead to insecurity because of the assumed security of the applied signature mechanism. One avenue of attack that still has to be considered is the possible replay of valid messages by the attacker, leading to, e.g., possible re-execution of previous remote calls. This needs to be mitigated by protocol, application or OS mechanisms, the details of which are beyond the scope of this paper. Also, routing information in the unencrypted frame header of messages will be disclosed to an attacker. This is not a problem, except in cases where messaging patterns as a side-channel disclose private information [5]. These can be mitigated through side-channel-free implementations of critical algorithms or changes to the hardware that prevent disclosure of routing information.

3.2 Space, Communication and Synchronity

To evaluate potential gains from employing T-IBE-T over other conventional methods described above, it is useful to compare the asymptotic complexity in space and the number of NoC frames in a communication sequence of Alice sending a one-frame payload to Bob in any combination of n partners. Table 2 provides an overview of the described key exchanges' properties.

In a symmetric encryption system, for arbitrary communication between n participants, the number of necessary keys is $O(n^2)$. However, if the key directory is not globally stored, each single participant will only need $O(n)$ keys for its $n-1$ possible partners.

In an asymmetric system that uses a simple RSA key exchange there are only $O(n)$ keys globally for the n possible recipients in the system. This means that a global key directory will be smaller than in the symmetric case. However, both in the symmetric and the RSA case, querying the global key directory will take a network roundtrip to the key directory, after which the obtained key can be used to send a posted communication to the recipient. With a local key directory, the directory query roundtrip will, of course, be avoided; therefore a posted communication can be sent immediately after the cheaper local lookup. Concerning network communications, this immediate ability to send also allows for asynchronous communication. One may even avoid all key directories with an RSA exchange by Alice asking Bob for his key, which must be signed by a trusted third party. However, asking Bob for his key will take one round-trip before Alice can encrypt a payload.

A system that uses a DH key exchange will, because of the necessary establishment of DH parameters for Alice and Bob, need one packet back and forth until a common key is known to Alice. Only then, in the third packet can Alice then send the intended payload to Bob. Conversely, in an IBE system like T-IBE-T, no directory of keys is needed whatsoever. Only Bob's routing address which is his *ID* needs to be known, however, this information is considered to be necessary and available in all cases mentioned. Bob's public key is his *ID*, which is immediately available to Alice, therefore a posted message with an encrypted payload may be sent immediately in the first frame, also allowing for asynchronous communication.

3.3 Security Tradeoffs

T-IBE-T intentionally is not intended to be resistant against more powerful adversaries such as full man-in-the-middle adversaries. If resistance against such adversaries is a goal, other methods should be used, e.g. signed DH exchanges. However, these come at a cost in network frames and forced sychronity. We also think that such adversaries can fully be excluded by design of the NoC hardware.

3.4 Key Escrow

A common critique of IBE systems is that the PKGS is able to decrypt all communication through its ability to compute the private key for any given address (key escrow). For a public, general-purpose communication system such as email, key escrow is highly undesirable from a user's point of view.

The circumstances in case of T-IBE-T, however, are different: First, the OS which controls the execution environment and protection mechanisms will be able to obtain and manipulate application data at will (this could, with appropriate hardware mechanisms such as secure enclaves [8] be avoided, an in-depth discussion is out of the scope of this paper however). Through T-IBE-T the OS would therefore not gain any additional capabilities. What is more, debugging and tracing is greatly aided by the ability of a developer or system administrator to decrypt NoC traffic in live captures as well as recordings. In contrast to computer networks, accessing the unencrypted data on the endpoints or using decrypting proxies is rather hard to impossible in NoC scenarios. Therefore we consider the key escrow functionality inherent to most IBE implementations beneficial, accounting for the special use case which T-IBE-T is designed for.

4 RELATED AND FUTURE WORK

Sharma et al. describe a two-step key agreement protocol for group communication keys in an MPSoC setting similar to ours [14]. Yet the higher number of steps in the agreement and different communication paradigm (group communication) cause us to see [14] as a possible complement to T-IBE-T. Similarly, Sepulveda et al. in [10–12] propose zone-based approaches or tunnels, and also use either predistribution of keys or DH-type exchanges for MPSoC communication, which are less advantageous than T-IBE-T in our evaluation scenario.

Besides the canonical example of email encryption, IBE has been proposed for uses such as IoT and body sensor networks [15, 16] and cloud computing [9]. TinyIBE as discussed in [15] is especially interesting for this work, since it also demonstrates the feasibility of IBE in performance-critical, resource-constrained systems. A special emphasis is placed on properly supporting the different classes of sensor nodes which may have different local resources and communication capabilities. Conversely in the T-IBE-T scenario, nodes are mostly similar, however applications and system services also participate in communications. Furthermore TinyIBE deals with an attacker who also steals and analyzes sensor nodes to obtain cryptographic keys, which in a SoC scenario, such as T-IBE-T, would involve the attacker stealing the whole system, a wholly different problem. However, in certain aspects, TinyIBE's thieving attacker may be compared to the T-IBE-T attacker compromising a tile. While TinyIBE also employs a setup procedure, circumstances in the T-IBE-T MPSoC bootup phase are very different. Overall, while TinyIBE shares many aspects, it needs to be adapted to fit the T-IBE-T scenario.

The current early idea of T-IBE-T is not yet implemented in a prototype software which is the obvious next step: We first plan to use e.g. Stanford PBC [6, 7] or TinyIBE [15] in a first prototype that would run on a network of off-the-shelf computers, because implementation is more straightforward and a node count of 1000 is realistically possible. This first prototype would be used to verify the relative costs of T-IBE-T compared with other techniques as described in the evaluation and gather experience on the implementation and use of T-IBE-T.

A second prototype would then be implemented in the actual target architecture called *invasive architecture*: The invasive architecture [17] is a tiled architecture linked by the invasive network-on-chip (*i*NoC) where very lightweight tasks called *i*lets can be cheaply created and executed on local and remote multicore tiles.

Table 2: Comparison of space, communication costs between Alice, Trent (global key directory or CA) and Bob, and synchronicity for key exchange methods described in this work. The ✓ symbol signifies a desirable, and ✘ an inferior property.

	symmetric		RSA		DH+RSA	T-IBE-T
key dist.	global dir.	local dir.	local dir.	CA	CA	IBE
key dir. size	$O(n^2)$ ✘	$O(n)$ ✘	$O(n)$ ✘	$O(1)$ ✓	$O(1)$ ✓	$O(1)$ ✓
synchronicity	sync ✘	async ✓	async ✓	sync ✘	sync ✘	async ✓
# frames A ↔ T	2 ✘	0 ✓	0 ✓	0 ✓	0 ✓	0 ✓
# frames A ↔ B	1 ✓	1 ✓	1 ✓	3 ✘	3 ✘	1 ✓

While current hardware prototypes do not reach the envisioned tile and core counts in the order of a thousand or more, this second prototype of T-IBE-T will be used to demonstrate the viability of the described setup and boot process as well as the incorporation of T-IBE-T as part of the invasive runtime support system's (*i*RTSS's) and operating system's (OctoPOS's) mechanisms. In both prototypes, the impact of T-IBE-T on existing applications will be evaluated.

A possible extension to T-IBE-T is the use of hierarchical IBE, where there is not a single PKGS that issues all private keys. Instead, for an *ID* x, the owner of the private key for that *ID* x can issue private keys for all *ID*s that have x as a prefix, e.g. xy. This means that the *ID* –which in T-IBE-T currently equals a routing address– would also need to represent the hierarchy of responsibilities as the prefix tree of addresses. The first question to be investigated is whether such a representation is sensible and practical.

Lastly, the T-IBE-T idea needs to be condensed into a protocol definition, incorporating experiences from the aforementioned prototypes that can then be formally evaluated, e.g. by proving desirable security properties.

5 CONCLUSION

This paper introduced T-IBE-T, an innovative approach to secure NoC communications in MPSoC systems through the use of identity-based encryption (IBE). T-IBE-T allows, other than with commonly used key exchange or key distribution mechanisms such as RSA, DH or Kerberos, for asynchronous, into-the-future and posted encrypted messaging. T-IBE-T enables micro-parallelism through efficient posted tile-to-tile communication, without needing expensive key exchange roundtrips, key tables or large centralized key directories. While in other contexts IBE key escrow is seen as problematic, MPSoC operations provides an area where key escrow enables better system monitoring and debugging.

ACKNOWLEDGMENTS

The authors would like to thank Peter Wägemann for providing helpful advice and proverbial occipital-percussive support.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is partially supported by the German Research Foundation (DFG) in the CRC / TRR 89 (Invasive Computing) subprojects C5 & C1 (https://invasic.cs.fau.de/).

REFERENCES

[1] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. 2005. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Advances in Cryptology – EUROCRYPT 2005*. Vol. 3494. Springer Berlin Heidelberg, Berlin, Heidelberg, 440–456. https://doi.org/10.1007/11426639_26

[2] Dan Boneh and Matt Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology — CRYPTO 2001*. Springer Berlin Heidelberg, Berlin, Heidelberg, 213–229.

[3] Sherman S. M. Chow. 2009. Removing Escrow from Identity-Based Encryption. In *Public Key Cryptography – PKC 2009 (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, 256–276.

[4] Eike Kiltz and Yevgeniy Vahlis. 2008. CCA2 Secure IBE: Standard Model Efficiency through Authenticated Symmetric Encryption. In *Topics in Cryptology – CT-RSA 2008*. Springer Berlin Heidelberg, Berlin, Heidelberg, 221–238.

[5] Paul C. Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology — CRYPTO '96 (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, 104–113.

[6] Ben Lynn. [n. d.]. The PBC (Pairing-Based Cryptography) library. https://crypto.stanford.edu/pbc/

[7] Ben Lynn. 2007. *On the Implementation of Pairing-based Cryptosystems*. Stanford University.

[8] Pieter Maene, Johannes Götzfried, Ruan De Clercq, Tilo Müller, Felix Freiling, and Ingrid Verbauwhede. 2017. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Comput.* PP, 99 (2017). https://doi.org/10.1109/TC.2017.2647955

[9] C. Schridde, T. Dörnemann, E. Juhnke, B. Freisleben, and M. Smith. 2010. An identity-based security infrastructure for Cloud environments. In *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*. 644–649. https://doi.org/10.1109/WCINS.2010.5541859

[10] Johanna Sepulveda, Daniel Flórez, Vincent Immler, Guy Gogniat, and Georg Sigl. 2017. Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs. *Microprocessors and Microsystems* 50 (May 2017), 164–174. https://doi.org/10.1016/j.micpro.2017.03.002

[11] Johanna Sepulveda, Daniel Flórez, Vincent Immler, Guy Gogniat, and Georg Sigl. 2017. Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs. *Microprocessors and Microsystems* 50 (2017), 164 – 174. https://doi.org/10.1016/j.micpro.2017.03.002

[12] Johanna Sepúlveda, Andreas Zankl, Daniel Flórez, and Georg Sigl. 2017. Towards Protected MPSoC Communication for Information Protection against a Malicious NoC. *Procedia Computer Science* 108 (Jan. 2017), 1103–1112. https://doi.org/10.1016/j.procs.2017.05.139

[13] Adi Shamir. 1985. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology (Lecture Notes in Computer Science)*. Springer Berlin Heidelberg, 47–53.

[14] G. Sharma, V. Kuchta, R. A. Sahu, S. Ellinidou, O. Markowitch, and J. Dricot. 2018. A Twofold Group Key Agreement Protocol for NoC based MPSoCs. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. 1–2. https://doi.org/10.1109/PST.2018.8514117

[15] P. Szczechowiak and M. Collier. 2009. TinyIBE: Identity-based encryption for heterogeneous sensor networks. In *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. 319–354. https://doi.org/10.1109/ISSNIP.2009.5416743

[16] Chiu C. Tan, Haodong Wang, Sheng Zhong, and Qun Li. 2008. Body Sensor Network Security: An Identity-based Cryptography Approach. In *Proceedings of the First ACM Conference on Wireless Network Security (WiSec '08)*. ACM, New York, NY, USA, 148–153. https://doi.org/10.1145/1352533.1352557

[17] Jürgen Teich, Jörg Henkel, Andreas Herkersdorf, Doris Schmitt-Landsiedel, Wolfgang Schröder-Preikschat, and Gregor Snelting. 2011. Invasive Computing: An Overview. In *Multiprocessor System-on-Chip*. Springer New York, New York, NY, 241–268. https://doi.org/10.1007/978-1-4419-6460-1_11