MATTER: A Wide-Block Tweakable Block Cipher

Roberto Avanzi\textsuperscript{1}, Orr Dunkelman\textsuperscript{2} and Kazuhiko Minematsu\textsuperscript{3}

\textsuperscript{1} Caesarea Rothschild Institute, University of Haifa, Haifa, Israel  
\texttt{roberto.avanzi@gmail.com}
\textsuperscript{2} Computer Science Department, University of Haifa, Haifa, Israel  
\texttt{orrd@cs.haifa.ac.il}
\textsuperscript{3} NEC Corporation, Kawasaki, Japan  
\texttt{k-minematsu@nec.com}

\textbf{Abstract.} In this note, we introduce the MATTER Tweakable Block Cipher, designed principally for low latency in low-area hardware implementations, but that can also be implemented in an efficient and compact way in software. MATTER is a 512-bit wide balanced Feistel network with three to six rounds, using the ASCON permutation as the round function. The Feistel network defines a keyed, non-tweakable core, which is made tweakable by using the encryption of the tweak as its key. Key and tweak are 320-bit inputs. MATTER is particularly suitable for use in an OCB-like mode of operation, with an encrypted checksum for authentication.

\textbf{Keywords:} Tweakable Block Ciphers · Lightweight Cryptography · Wide-Block Ciphers · Memory Encryption

1 Introduction

NIST has recently shown interest in standardizing tweakable, wide-block block ciphers, focusing on “accordion” ciphers based on the AES [DR02]. Software implementations of such a design can be efficient if the Instruction Set Architecture (ISA) exposes AES instructions. In hardware, however, the AES can lead to large or slow implementations — this is one of the reasons a large amount of research has been poured into lightweight primitives during the last several years, for instance CLEFIA [SSA+07], ChaCha20 [Ber08], KATAN and KTANTAN [CDK09], KLEIN [GNL11], LED [GPPR11], PRESENT [BKl+07], PRICE [BCG+12], SIMON and SPECK [BSS+13], MIDORI [BBI+15], QARMA [Ava17], SKINNY and MANTIS [BJK+16], ASCON [DEMS21], BipBip [BDD+23], QARMAv2 [ABD+23], and many other ciphers.

In this note, we propose a solution that prioritizes hardware implementations, based on ASCON, which is set to be standardized by NIST. Since ASCON has been designed to be efficient not only on high-end CPUs but also on resource-constrained devices, both in hardware and in software, our proposal targets the same use cases as well.

The initial motivation for the present work is memory encryption. The security model is straightforward: hardware inside the physical perimeter of a System-on-a-Chip (SoC), including the memory controllers, is trusted, while the external memory bus and memory itself are untrusted. We focus mainly on memory confidentiality where memory is encrypted in Cache Line (CL) sized granules, but also briefly touch integrity and modes of operation.

There are two main approaches to memory encryption: In the first approach, direct encryption, clear data is input to a block cipher and the output is written to external memory. A common requirement is that encryption provides spatial uniqueness, meaning
the same plaintext at different memory locations produces different ciphertexts. This can be achieved using a **Tweakable Block Cipher** (TBC) [LRW02, LRW11], i.e., a block cipher with three inputs: the secret key, a text, and a **tweak**. The permutation computed by the cipher depends on the tweak as unpredictably as it does on the key. An adversary, however, is able to control the tweak but cannot use this capability to help recover the key. This makes the memory address suitable as a tweak. In fact, one of the earliest uses of TBCs was memory encryption [HT13].

The second approach is suitable if **temporal uniqueness** is required, meaning repeated writes of the same plaintext to the same location produce different ciphertexts. A keystream is generated by a block cipher or a hash function in counter mode, or a stream cipher, and is XOR-ed with the plaintext to produce the ciphertext. The keystream generator uses a secret key, the memory address, and a counter or nonce as inputs. Each CL has its own counter or nonce, which is refreshed before each memory write.

Keystream-based encryption keeps the cipher off the critical path between CPU and external memory, reducing the additional read latency to a single XOR. However, the nonces must be stored in RAM, reducing the amount of the latter available to applications and increasing memory traffic. This can lead to severe performance degradation [AMS+22].

Another problem with the second approach is ciphertext malleability, as flipping a bit in the ciphertext flips the corresponding bit in the plaintext. This enables, for instance, RowHammer attacks [KDK+14], unless integrity tags are generated and verified. These tags also use memory space, adding pressure on the memory subsystem.

Direct encryption, even in the absence of authentication, partially mitigates RowHammer attacks, because any ciphertext change makes its decryption uncorrelated with the original plaintext. This and the absence of ciphertext expansion make direct encryption attractive for practical deployments. This note deals with direct encryption.

**ASCON** is a 320-bit permutation, and common CL sizes (such as 512 or 1024 bits) cannot be partitioned into 320-bit blocks, preventing its use for direct encryption of CLs. **ASCON** is also not inherently keyed or tweakable. Our construction addresses these issues.

### Outline of the Paper
In Section 2 we define the non-tweakable core of **MATTER** and in Section 3 we convert it into a TBC. The derivation of the round keys is described in Section 4. In Section 5 we present suitable modes of operation, covering memory encryption and general-purpose use. We sketch provable security in Section 6. Implementation is briefly discussed in Section 7. Finally, in Section 8 we conclude with open questions.

## 2 Definition of the non-tweakable (core) version of **MATTER**

The block cipher **MATTER** is a 512-bit balanced Feistel network [Sor84, LR85, LR88]. It is designed around a non-tweakable, keyed core function, aptly named **MATTER**\(_\text{core}\), and a process for deriving its key from a main key and a tweak. **MATTER**\(_\text{core}\) comes in versions determined by the number of Feistel rounds, between three and six. Its Feistel function is represented in Figure 1, and consists of the following operations, in the given order:

1. Expansion of the input from 256 to 320 bits by padding it with zero;
2. Addition of a round key \(k_i\) (this round key is 320 bits long);
3. An application of **ASCON**\(_r\)-\(p^r\), where the notation **ASCON**\(_r\)-\(p^r\) denotes the \(r\)-round **ASCON** permutation;
4. Truncation of the result to 256 bits; and
5. Addition of a round key \(k'_i\) (this round key needs to be only 256 bits).
We restrict $r_i \geq 4$ to guarantee full diffusion in the ASCON function and the number of Feistel rounds is limited to six since we want the total latency of the core function not to exceed twice that of ASCON-p^{12} — in other words, 24 ASCON rounds — and a few XORs. The function $F_r$ is computed on one branch and the result added to the other branch, then the two branches are swapped. This operation is performed three to six times. We write $\text{MATTER}^{\text{core}}_{r_0, r_1, r_2}$ for the three Feistel rounds version, $\text{MATTER}^{\text{core}}_{r_0, r_1, r_2, r_3}$ for the four Feistel rounds version, and so on.

Regardless of the claimed security level associated with a parameter set, keys are always understood as 320-bit strings. This approach mirrors the methodology in BipBip [BDD+23], where a 256-bit main key is used even if the claimed security level is 96 bits, and dispenses with the need for a potentially costly key schedule.

Shorter keys can of course be used and they can be just zero padded, but we strongly recommend the use of a secure Key Derivation Function (KDF) to extend short keys to 320 bits. This can be useful in the case the use of separate keys is mandated by architecture, system design, or regulation, as shorter reduce keys storage requirements.

3 Definition of the tweakable version of MATTER

Our approach is inspired by the Masked Even-Mansour Masked Even-Mansour (MEM) construction [GJMN16]. In this method, the round keys are not directly derived from the main key, but from a core key which is the encryption of the tweak using the main key. The tweak encryption function is a XOR, Encrypt, and XOR (XEX) construction [Rog04].

If $K$ and $T$ are the 320-bit key and tweak, then $k = \text{ASCON}_{r'}(K \oplus T) \oplus K$, for some positive integer $r'$. The notation $\text{MATTER}^{r'}_{r_0, r_1, r_2, \ldots}$ denotes the tweakable block cipher obtained by adding tweak encryption to $\text{MATTER}^{\text{core}}_{r_0, r_1, r_2, \ldots}$.

4 Derivation of the round keys

Put $k = K$, resp., $k = \text{ASCON}_{r'}(K \oplus T) \oplus K$ for the non-tweakable, resp., tweakable version of the cipher. The round keys $k_i$ are given by $k_i = (2^i \cdot k) \oplus c_i$, where 2 represents the image of $x$ in $\mathbb{F}_{2^{320}} = \frac{\mathbb{F}_2[x]}{x^{320} + x^{256} + x^{224} + x^{192} + x^{128} + x^6 + 1}$. This is an inexpensive operation — a shift and four single-bit XORs — and therefore it does not impact latency upon decryption (cf. Section 7). It is $c_0 = 0$ and for $i > 0$ the values $c_i$ are 64-bit constants, padded with zeros, taken from the hexadecimal expansion of the fractional part of $\pi$:

$c_1 = 13198A2E03707344$,  $c_2 = A4093822299F31D0$,  $c_3 = 082EFA98EC4E6C89$,  $c_4 = 452821E638D01377$,  and $c_5 = BE5466CF34E90C6C$. 
The round keys $k'_i$ are defined by $k'_i = \tau_{256}(k_i)$, where $\tau_s$ is the function that retains only the $s$ least significant bits of its input. The 64 most significant bits of $k_i$ are not used to determine $k'_i$. The 256 least significant bits of the difference between $k'_i$ and $k_{i+1}$ is the truncation of $3 \cdot k_i$ (3 represents $x + 1$) and therefore there are no fixed cancellation relations between $k'_i$ and $k_{i+1}$.

5 Integrity Tags and Modes of Operation

5.1 Memory Protection

MATTER encrypts 512-bit blocks, so, in the context of encryption of CLs, we only consider CLs whose lengths are a multiple of 512 bits.

Let us first discuss encryption only. In the context of direct encryption, avoiding ciphertext expansion is desirable, otherwise a keystream could be used. For CLs longer than 512 bits, once the first block has been encrypted by MATTER and thus the first core key $k$ has been computed, each further block is encrypted by using MATTER core, with each core key derived by multiplying the previous one by, say, 5 (i.e. $x^2 + 1$) in $F_{2520}$. Other field elements can be used, as long as they are not a power of 2 or 3 times a power of 2, to avoid reusing the round key differences mentioned in Section 4.

Let us now discuss the case where a tag is generated. If the length of a CL is equal to 512 bits, a tweaked encryption of the 64 or 128 least significant bits of the CL is used as a tag, possibly truncated to a shorter length. The tweak must contain the physical memory address of the individual 64-bit or 128-bit ciphertext blocks. This way of computing the tag does not only provide authenticity, but also allows error detection and correction of errors up to few bits [JLK+23].

Note that replay attacks are possible unless the tags are themselves protected.

The integrity key must be generated independently of the encryption key. It is a common requirement that memory regions belonging to different processes have different encryption keys. However, a single integrity key for the system is sufficient, as the tag, viewed as a function of the ciphertext, depends on the encryption key.

5.2 General Purpose Usage

For conciseness, we do not present a full description of a general purpose Authenticated Encryption with Associated Data (AEAD) mode of encryption based on MATTER, but we
outline the key points. The mode is a variant of Flat-\textcolor{red}{\Theta}\textcolor{red}{\textcolor{red}{C}}B \cite{IMO}, itself derived from \textcolor{red}{\Theta}CB, and its cost is one encryption per block, plus one encryption to finalize the tag.

When using \textcolor{red}{\textit{MATTER}} with variable length messages, the tweak for the first 512-bit block contains a unique \textit{Initialization Vector} (IV). Once this tweak is encrypted to obtain the first core key, the successive ones can be obtained in one of two ways: either encrypting successive values of the tweak, or by multiplying the first encrypted tweak repeatedly by some element of $\mathbb{F}_{2^{320}}$, say 5, as in the memory encryption case. For the final partial block, ciphertext stealing \cite{Dwo11} (the method can be traced back to \cite[p. 78]{MM}) is used.

An alternative to ciphertext stealing is to treat \textcolor{red}{\textit{MATTER}} as format preserving encryption, i.e., for a $b$-bit final partial block, to have two branches of $b_t = \lfloor b/2 \rfloor$ and $b_r = \lceil b/2 \rceil$ bits. This can be obtained by simply masking to the intermediate results of the 512-bit wide \textcolor{red}{\textit{MATTER}} function, that is the inputs and the compression function. This requires $b > 1$ (this is usually not problem as the bit-lengths of messages are often multiples of 8). Domain separation is necessary when this happens.

To compute an $s$-bit tag, an encrypted checksum of the plaintext is used, similarly to \textcolor{red}{\Theta}CB \cite{Rog04}, where the total length of the message is included in the tweak of the final encryption, together with domain separation. Generalizing the approach in \cite{IM}, we generate the tags by taking the 64, 128, or 256 least significant bits of each 512-bit plaintext block, adding these values, and encrypting the result including the length of the message in the tweak (for instance, using \textcolor{red}{\textit{QARMaV2}} for the smaller tags, but many choices are available including \textcolor{red}{\textit{MATTER}} itself).

In presence of Associated Data (AD) $A$, the contributions to the tag from $A$ are computed on an encryption of the latter. There is no need to use ciphertext stealing or a special format preserving version of \textcolor{red}{\textit{MATTER}} to encrypt the last block, if fractional, of the AD: The last block is simply padded using the string $10^s$, if necessary, before encryption, with the length of $A$ included in the tweak with domain separation. Then, the sum of the 64, 128, or 256 least significant bits of each 512-bit encrypted block is saved, to be added to the encrypted checksum of the public part to form the tag.

6 Security

Assume that \textcolor{red}{\textit{ASCON}} with sufficiently many rounds (e.g., 12 or more) is used, to ensure it can be considered a PRP in an Even-Mansour \cite{EM} construction.

Provable security analysis of \textcolor{red}{\textit{MATTER}} follows known results, assuming the \textcolor{red}{\textit{ASCON}} permutation is ideal. For simplicity we sketch the proof for non-tweakable case, with four Feistel rounds. Let $q$ be the number of encryption or decryption queries of 512-bit messages, and $p$ be the number of primitive queries to \textcolor{red}{\textit{ASCON-p}}.

First, using a hybrid argument on the round function, similar to MEM, we get \textcolor{red}{\textit{ASCON}} using four independent 320-bit random permutations with expansion and truncation (for 256-bit 1/O) at a cost of $q \cdot p/2^{320} + q^2/2^{320}$. Applying the PRP-PRF switching lemma (cf. \cite{BR} and references therein) brings an additional cost of $q^2/2^{320}$. Thus, \textcolor{red}{\textit{MATTER}} with four independent 256-bit random functions with a distinguishing advantage of $q^2/9^8/2^{256}$.

The total distinguishing advantage is $O(q \cdot p/2^{320} + q^2/2^{256})$.

Improvements are possible. Using the classical result of Naor and Reingold \cite{NR} instead of the original arguments of Luby-Rackoff, the top and bottom only need to be almost XOR-universal hash functions, relaxing the first hybrid argument, i.e., allowing fewer \textcolor{red}{\textit{ASCON}} rounds for the top and bottom Feistel functions. This permits constructions such as \textcolor{red}{\textit{MATTER}}_{6,12,6} in place of \textcolor{red}{\textit{MATTER}}_{8,8,8}, with the same latency but possibly better security. Advanced constructions such as \cite{GW, CLL} can reduce the number of independent keys. However, the main bottleneck would be a collision on 256-bit branches, meaning 128-bit security.

Assuming $p \ll 2^{64} q$, the first summand in the distinguishing advantage is negligible.
Table 1: Comparison of Direct Memory Encryption Primitives in TSMC 5nm Process.

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Width</th>
<th>Rounds</th>
<th>Area (µm²)</th>
<th>GE ps</th>
<th>Delay (µm²)</th>
<th>GE ps</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>128</td>
<td>10</td>
<td>2304.1</td>
<td>2883</td>
<td>3064</td>
<td></td>
</tr>
<tr>
<td>AES-128/XEX (write)</td>
<td>128</td>
<td>10 + 10</td>
<td>4688.3</td>
<td>5875</td>
<td>6156</td>
<td>3092</td>
</tr>
<tr>
<td>AES-128/XEX (read)</td>
<td>128</td>
<td>12</td>
<td>2635.4</td>
<td>33025</td>
<td>3686</td>
<td></td>
</tr>
<tr>
<td>AES-192</td>
<td>128</td>
<td>12</td>
<td>5352.6</td>
<td>67074</td>
<td>7400</td>
<td>3713</td>
</tr>
<tr>
<td>AES-192/XEX (write)</td>
<td>128</td>
<td>12 + 12</td>
<td>3238.7</td>
<td>40585</td>
<td>4290</td>
<td></td>
</tr>
<tr>
<td>AES-192/XEX (read)</td>
<td>128</td>
<td>14</td>
<td>6559.2</td>
<td>82194</td>
<td>8607</td>
<td>4316</td>
</tr>
<tr>
<td>AES-256</td>
<td>128</td>
<td>14</td>
<td>1635.6</td>
<td>20496</td>
<td>1561</td>
<td></td>
</tr>
<tr>
<td>AES-256/XEX (write)</td>
<td>128</td>
<td>14 + 14</td>
<td>1620.3</td>
<td>20305</td>
<td>1409</td>
<td></td>
</tr>
<tr>
<td>AES-256/XEX (read)</td>
<td>128</td>
<td>20</td>
<td>1893.5</td>
<td>23727</td>
<td>1645</td>
<td></td>
</tr>
<tr>
<td>QARMA-128 (r = 11)</td>
<td>128</td>
<td>24</td>
<td>2166.8</td>
<td>27152</td>
<td>1879</td>
<td></td>
</tr>
<tr>
<td>QARMAv2-128 (r = 11)</td>
<td>128</td>
<td>24</td>
<td>2166.8</td>
<td>27152</td>
<td>1879</td>
<td></td>
</tr>
<tr>
<td>QARMAv2-128 (r = 13)</td>
<td>128</td>
<td>28</td>
<td>2166.8</td>
<td>27152</td>
<td>1879</td>
<td></td>
</tr>
<tr>
<td>QARMAv2-128-256 (r = 15)</td>
<td>128</td>
<td>32</td>
<td>2166.8</td>
<td>27152</td>
<td>1879</td>
<td></td>
</tr>
<tr>
<td>ASCON-p^{12}</td>
<td>320</td>
<td>12</td>
<td>2228.3</td>
<td>27923</td>
<td>826</td>
<td></td>
</tr>
<tr>
<td>MATT{8,8,8}</td>
<td>512</td>
<td>8 + 24</td>
<td>6309.9</td>
<td>79069</td>
<td>2299</td>
<td>1724</td>
</tr>
<tr>
<td>MATT{6,6,6,6}</td>
<td>512</td>
<td>6 + 24</td>
<td>6040.4</td>
<td>75567</td>
<td>2085</td>
<td>1668</td>
</tr>
</tbody>
</table>

Compared to the second. Thus, with memory for \( q \) plaintext/ciphertext pairs, the advantage is roughly \( q^2 / 2^{256} \), requiring \( \propto 2^{256}/q^2 \) attempts to succeed. The total time is dominated by the invocations to the encryption and decryption oracles, as well as to the ASCON permutation, so we have approximately \( 2^{256}/q \) data and time with \( q \) memory. Note that the memory requirements and running time may increase if \( p > q \).

Now, distinguishability often leads to key-recovery attacks, with usually comparable complexity. This leads to a memory-time tradeoff \( M \cdot T \propto 2^{256} \).

7 Implementations

In Table 1 we pit MATT{8,8,8} and MATT{6,6,6,6} against two other families of choices of block ciphers for memory encryption in a direct mode: the AES and QARMA-128/QARMAv2-128. These two particular instantiations of MATTER have area and latency similar to other options such as, say, MATT{4,4,4,4,4,4}. The actual security level of these instantiations of MATTER has not been determined yet through cryptanalysis, but we expect that it will be above 128 bits and approach 256 bits in the memory-time tradeoff sense of Section 6.

For the ciphers we either report implementation results in a low-voltage TSMC 5nm lithography with the \texttt{tsmc\_sch280pp57\_clm05fb41001} library, taken from [ABD+23] or extrapolated, using the known area and latency values for XOR gates and intermediate registers in the given process. Area and latency are reported for both area optimized and latency optimized implementations.

Two delay (i.e., additional latency) values are reported for the XEX construction and for MATTER, namely for memory write and read operations. The reason is that when a memory read request is issued, the computation of the encrypted tweak can almost always be completed while the memory controller is waiting for the requested data to reach it. Therefore, in both designs only the encryption latency is on the critical path for memory reads. But, on memory writes also the tweak encryption must always be taken.
into account.

Fully unrolled AES-XEX can approach the area of MATTER, but with worse latency. Using a single fully unrolled AES instance twice almost halves the area with a minor latency increase. However, since MATTER is 512 bits wide and AES is 128 bits wide, AES-XEX must be invoked four times, either as monolithic circuit or in a pipelined implementation: both solutions negatively affect latency and area. Replicating the circuit to process four blocks simultaneously makes the AES-based solution significantly more expensive in terms of area, but the latency remains much higher than MATTER.

QARMA and QARMAv2 are significantly lighter on resources, with similar latency to MATTER and lower area. However, as with the AES, invoking the circuit four times or pipelining it increases latency, and replicating it four times makes the area too large, unless the smallest 24-round version is used, in which cases the latter’s area is comparable to MATTER.

For software implementations, MATTER inherits the most important advantages of ASCON [DEM21, Section 7.2], such as suitability to bit-sliced implementations and compact code, making it ideal for restricted environments.

8 (Temporary) Conclusion: Open Questions

While we can expect that MATTER_{12,12,12} and MATTER_{12,12,12,12} offer high security levels — say at 256 bits of memory-time product as in Section 6 — they have large area and high latency. To reduce latency, we can lower the number of rounds in ASCON, for example, from 12 to 8 in the three Feistel rounds version and even to 6 in the four Feistel rounds version. For five- or six Feistel round versions, we can go as low as 6 rounds in ASCON, keeping the core latency about twice that of the ASCON-p_{12} permutation.

The question is however whether we can achieve a desired level bit-security for specific instances such as MATTER_{8,8,8}, MATTER_{6,12,6}, MATTER_{6,6,6,6}, or even other variants such as MATTER_{8,8,8,4} and MATTER_{4,4,4,4,4} or MATTER_{8,4,4,4,4,4}. In particular, an important goal is to find lower bounds for the classical time complexity when at most 2^{64}, 2^{80}, or 2^{96} data is available, first for the non-tweakable core and then for the entire TBC. Ad-hoc cryptanalysis is required, and it will be the subject of future investigations.

References


MATTER: A Wide-Block Tweakable Block Cipher


MATTER: A Wide-Block Tweakable Block Cipher


