Hardness of Learning AES Key

Sultan Nurmukhamedov[1], Artur Pak[1], Rustem Takhanov[1] and Zhenisbek Assylbekov[1,*]

[1]Deparment of Mathematics, Nazarbayev University, 53 Kabanbay Batyr ave., Astana, Kazakhstan, 010000

Abstract

We show hardness of learning AES key from pairs of ciphertexts under the assumption of *computational* closeness of AES to pairwise independence. The latter is motivated by the recent result of Liu et al. [1].

Keywords

AES, Machine Learning, Computational Hardness

1. Introduction and Main Result

Advanced Encryption Standard (AES) is one of the most popular encryption algorithms today. It underlies the TLS 1.3 protocol, which is used by most modern websites, email services, instant messengers, etc. The US National Security Agency uses AES to encrypt materials classified as top secret.[1] It would seem that with such a wide distribution there should be a strong guarantee of the security of this algorithm. However, at the moment, results on the provable security of AES against various cryptanalysis methods are scarce. This is primarily due to the fact that AES is *not* based on any mathematically hard problem. On the contrary, this algorithm is a heuristic proposed by Daemen and Rijmen [2] in the late 90s. It is noteworthy that since then no one has managed to build a successful attack on the AES. State of the art attacks are only marginally better than brute force: for example, Tao and Wu [3]'s biclique attack requires 2^{126} operations to recover a 128-bit AES key (compared to 2^{128} operations with a brute force attack).

This lack of computationally feasible attacks on AES suggests that this method is indeed secure, but as we noted above, we currently have very little understanding of its provable security. Here we would like to highlight the recent work by Liu et al. [1], which shows for two different inputs the ϵ-closeness of the corresponding AES outputs to a uniform distribution under the randomness of its key. This property, also referred to as ϵ-closeness to pairwise independence, implies AES's resistance to linear and differential cryptanalysis. In this paper, motivated by the result of Liu et al. [1], we show the resistance of AES to attacks based on machine learning.

Let $F : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$ be a permutation family, denoted as $F_{\mathbf{k}}(\mathbf{x})$, where $\mathbf{k} \in \{0,1\}^m$

Discussion Papers - 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA 2022)

*Corresponding author.
sultan.nurmukhamedov@nu.edu.kz (S. Nurmukhamedov); artur.pak@nu.edu.kz (A. Pak); rustem.takhanov@nu.edu.kz (R. Takhanov); zhassylbekov@nu.edu.kz (Z. Assylbekov)
0000-0003-0095-9409 (Z. Assylbekov)

[1]https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/cnss15fs.pdf

is a key, and $\mathbf{x} \in \{0,1\}^n$ is an input. AES is a special case of F with $m \in \{128, 192, 256\}$ and $n = 128$. In this work, we prove the resistance of a permutation family F to attacks based on machine learning under the following

Assumption 1. *For a pair of distinct inputs* $\mathbf{x}$ *and* $\mathbf{x}'$*, and a uniformly sampled key* $\mathbf{k}$*, the distribution of the corresponding pair* $[F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')]$ *is computationally indistinguishable from the uniform distribution of two random distinct n-bit strings* $[\mathbf{u}, \mathbf{u}']$*, i.e. for any* poly$(n)$*-time algorithm* D

$$\left|\Pr_{\mathbf{k}}[D(F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')) = 1] - \Pr_{\mathbf{u},\mathbf{u}'}[D(\mathbf{u}, \mathbf{u}') = 1]\right| \leq 1/\text{poly}(n) \tag{1}$$

Note that the result of Liu et al. [1] differs from Assumption 1 in that we require only the initial key to be random, as is the case in the real AES.

We show that existence of a function computable in poly(n) time that, given a pair of arbitrary ciphertexts, can recover one of the keys consistent with those ciphertexts, would result in a polynomial distinguisher that contradicts Assumption 1. Our main result is the following

Theorem 1. *Let* $\mathbf{x}$ *and* $\mathbf{x}'$ *be arbitrary distinct n-bit strings and assume there exists a function* $h_{\mathbf{x},\mathbf{x}'} : \{0,1\}^{2n} \to \{0,1\}^n$ *such that*

$$h_{\mathbf{x},\mathbf{x}'}(\mathbf{y}, \mathbf{y}') = \begin{cases} \mathbf{k}, \text{ if } \exists \mathbf{k} : [F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')] = [\mathbf{y}, \mathbf{y}'] \\ \mathbf{0}, \text{ otherwise} \end{cases}, \tag{2}$$

and $h_{\mathbf{x},\mathbf{x}'}$ *is computable in* poly(n) *time. Then for a random uniform n-bit string* $\mathbf{k}$ *the distribution of* $[F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')]$ *is* computationally distinguishable *from that of two uniformly sampled distinct n-bit vectors.*

Remark. Under Assumption 1 there is no efficient learner for the class $\mathscr{H} := \{h_{\mathbf{x},\mathbf{x}'} \mid \mathbf{x}, \mathbf{x}' \in \{0,1\}^n, \ \mathbf{x} \neq \mathbf{x}'\}$, where each $h_{\mathbf{x},\mathbf{x}'}$ is given by (2). If there were such a learner, then by sampling uniformly at random $\ell = \text{poly}(n)$ keys $\{\mathbf{k}_i\}_{i=1}^{\ell}$, and computing $[F_{\mathbf{k}_i}(\mathbf{x}), F_{\mathbf{k}_i}(\mathbf{x}')]$, we could generate a labeled training sample of pairs $([F_{\mathbf{k}_i}(\mathbf{x}), F_{\mathbf{k}_i}(\mathbf{x}')], \mathbf{k}_i)$, which should suffice for our learner to figure out an (ϵ, δ) approximation (in PAC sense) of $h_{\mathbf{x},\mathbf{x}'}$, which by Theorem 1 would result in a polynomial time distinguisher that contradicts Assumption 1.

2. Proof of Theorem 1

Fix arbitrary distinct $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$, and let $h_{\mathbf{x},\mathbf{x}'}$ be defined by (2). Consider Algorithm 1, which we denote $D_{\mathbf{x},\mathbf{x}'}(\mathbf{y}, \mathbf{y}')$ for brevity. Randomly pick $\mathbf{k}$ from a uniform distribution over $\{0,1\}^n$. Feeding $F_{\mathbf{k}}(\mathbf{x})$, $F_{\mathbf{k}}(\mathbf{x}')$ as input to $D_{\mathbf{x},\mathbf{x}'}$, Line 1 produces $\boldsymbol{\kappa}$ such that $F_{\boldsymbol{\kappa}}(\mathbf{x}) = F_{\mathbf{k}}(\mathbf{x})$ and $F_{\boldsymbol{\kappa}}(\mathbf{x}') = F_{\mathbf{k}}(\mathbf{x}')$. Thus Lines 2&3 give us

$$\begin{aligned} \boldsymbol{\xi} &\leftarrow F_{\boldsymbol{\kappa}}^{-1}(F_{\mathbf{k}}(\mathbf{x})) = F_{\boldsymbol{\kappa}}^{-1}(F_{\boldsymbol{\kappa}}(\mathbf{x})) = \mathbf{x}. \\ \boldsymbol{\xi}' &\leftarrow F_{\boldsymbol{\kappa}}^{-1}(F_{\mathbf{k}}(\mathbf{x}')) = F_{\boldsymbol{\kappa}}^{-1}(F_{\boldsymbol{\kappa}}(\mathbf{x}')) = \mathbf{x}', \end{aligned}$$

and algorithm outputs 1. Hence

$$\Pr_{\mathbf{k}}\left[D_{\mathbf{x},\mathbf{x}'}(F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')) = 1\right] = 1 \tag{3}$$

Algorithm 1 Distinguisher

Input: $\mathbf{y}, \mathbf{y}' \in \{0,1\}^n$ s.t. $\mathbf{y} \neq \mathbf{y}'$
Parameter: $\mathbf{x}, \mathbf{x}' \in \{0,1\}^n$ s.t. $\mathbf{x} \neq \mathbf{x}'$

$\boldsymbol{\kappa} \leftarrow h_{\mathbf{x},\mathbf{x}'}(\mathbf{y}, \mathbf{y}')$
$\boldsymbol{\xi} \leftarrow F_{\boldsymbol{\kappa}}^{-1}(\mathbf{y})$
$\boldsymbol{\xi}' \leftarrow F_{\boldsymbol{\kappa}}^{-1}(\mathbf{y}')$
if $\boldsymbol{\xi} = \mathbf{x}$ **and** $\boldsymbol{\xi}' = \mathbf{x}'$ **then**
    return 1.
else
    return 0.
end if

Now randomly pick n-bit strings $\mathbf{u}, \mathbf{u}'$ without replacement from the uniform distribution over $\{0,1\}^n$ and feed them as input to $D_{\mathbf{x},\mathbf{x}'}$. Intuitively, in this case the event $A := \{h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') \neq \mathbf{0}\}$ has low probability. Let us upperbound the latter using the union bound:

$$\begin{aligned}
\Pr[A] &= \Pr_{\mathbf{u},\mathbf{u}'}[h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') \neq \mathbf{0}] \\
&= \Pr_{\mathbf{u},\mathbf{u}'}[\exists \boldsymbol{\kappa} \neq \mathbf{0} \ : \ [F_{\boldsymbol{\kappa}}(\mathbf{x}), F_{\boldsymbol{\kappa}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}']] \\
&= \Pr_{\mathbf{u},\mathbf{u}'}\left[\bigcup_{\boldsymbol{\kappa} \neq \mathbf{0}} [F_{\boldsymbol{\kappa}}(\mathbf{x}), F_{\boldsymbol{\kappa}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}'])\right] \\
&\leq \sum_{\boldsymbol{\kappa} \neq \mathbf{0}} \Pr_{\mathbf{u},\mathbf{u}'}[[F_{\boldsymbol{\kappa}}(\mathbf{x}), F_{\boldsymbol{\kappa}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}']] \qquad (4)
\end{aligned}$$

Notice that $[F_{\boldsymbol{\kappa}}(\mathbf{x}), F_{\boldsymbol{\kappa}}(\mathbf{x}')]$ is a fixed $2n$-bit string, and the joint p.d.f. of $\mathbf{u}, \mathbf{u}'$ has the form

$$\Pr_{\mathbf{u},\mathbf{u}'}(\mathbf{u} = \boldsymbol{v}, \mathbf{u}' = \boldsymbol{v}') = \frac{1}{2^n(2^n - 1)}, \qquad \boldsymbol{v} \neq \boldsymbol{v}'. \qquad (5)$$

Combining (4) and (5), we have

$$\Pr_{\mathbf{u},\mathbf{u}'}[A] \leq \sum_{\boldsymbol{\kappa} \neq \mathbf{0}} \frac{1}{2^n(2^n - 1)} = \frac{1}{2^n}. \qquad (6)$$

When $h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') = \boldsymbol{\kappa} \neq \mathbf{0}$, we have $F_{\boldsymbol{\kappa}}^{-1}(\mathbf{u}) = \mathbf{x}$, $F_{\boldsymbol{\kappa}}^{-1}(\mathbf{u}') = \mathbf{x}'$, and thus we can write

$$\Pr_{\mathbf{u},\mathbf{u}'}[D_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') = 1 \mid A] = 1 \qquad (7)$$

Now we turn to the event when $h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}')$ outputs the zero key. This happens if one of the following events occurs:

$$\begin{aligned}
B &:= \{h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') = \mathbf{0}\} \cap \{[F_{\mathbf{0}}(\mathbf{x}), F_{\mathbf{0}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}']\} \\
C &:= \{h_{\mathbf{x},\mathbf{x}'}(\mathbf{u}, \mathbf{u}') = \mathbf{0}\} \cap \{\nexists \boldsymbol{\kappa} \in \{0,1\}^n \ : \ [F_{\boldsymbol{\kappa}}(\mathbf{x}), F_{\boldsymbol{\kappa}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}']\}
\end{aligned}$$

By Eq. (5), we have

$$\Pr_{\mathbf{u},\mathbf{u}'}[B] \leq \Pr[[F_{\mathbf{0}}(\mathbf{x}), F_{\mathbf{0}}(\mathbf{x}')] = [\mathbf{u}, \mathbf{u}']] = \frac{1}{2^n(2^n - 1)}. \qquad (8)$$

In the event B, we have $[F_{\mathbf{0}}^{-1}(\mathbf{u}), F_{\mathbf{0}}^{-1}(\mathbf{u'})] = [\mathbf{x}, \mathbf{x'}]$, and thus Alg. 1 produces 1 in this case, i.e.

$$\Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1 \mid B] = 1. \tag{9}$$

In the event C, $h_S(\mathbf{u}, \mathbf{u'})$ outputs $\mathbf{0}$ which is *not* a key that maps $[\mathbf{x}, \mathbf{x'}]$ to $[\mathbf{u}, \mathbf{u'}]$ under AES, and we have

$$\begin{aligned}
&\Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1 \mid C] \\
&= \Pr_{\mathbf{u},\mathbf{u'}}[F_{\mathbf{0}}^{-1}(\mathbf{u}) = \mathbf{x}, F_{\mathbf{0}}^{-1}(\mathbf{u'}) = \mathbf{x'} \mid C] \\
&= \Pr_{\mathbf{u},\mathbf{u'}}[\mathbf{u} = F_{\mathbf{0}}(\mathbf{x}), \mathbf{u'} = F_{\mathbf{0}}(\mathbf{x'}) \mid C] = 0
\end{aligned} \tag{10}$$

Now we can decompose the probability that $D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'})$ outputs 1 as follows:

$$\begin{aligned}
&\Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1] \\
&= \Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1 \mid A] \cdot \Pr_{\mathbf{u},\mathbf{u'}}[A] \\
&+ \Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1 \mid B] \cdot \Pr_{\mathbf{u},\mathbf{u'}}[B] \\
&+ \Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1 \mid C] \cdot \Pr_{\mathbf{u},\mathbf{u'}}[C].
\end{aligned} \tag{11}$$

Plugging (6), (7), (9), (8), (10) into (11), we have

$$\begin{aligned}
&\Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1] \leq 1 \cdot \frac{1}{2^n} + 1 \cdot \frac{1}{2^n(2^n - 1)} + 0 \\
&= \frac{2^n - 1 + 1}{2^n(2^n - 1)} = \frac{1}{2^n - 1}.
\end{aligned} \tag{12}$$

Finally, combining (3) and (12), we get

$$\begin{aligned}
&\left| \Pr_{\mathbf{k}}\left[D_{\mathbf{x},\mathbf{x'}}(F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x'})) = 1\right] - \Pr_{\mathbf{u},\mathbf{u'}}[D_{\mathbf{x},\mathbf{x'}}(\mathbf{u}, \mathbf{u'}) = 1] \right| \\
&\geq 1 - \frac{1}{2^n - 1},
\end{aligned}$$

which means that Alg. 1 is a poly(n)-time distinguisher between the distribution of $[F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x'})]$ and the distribution of two distinct random n-bit strings, and this concludes the proof.

3. Empirical Verification of Assumption 1

Although Assumption 1 is motivated by the theoretical result of Liu et al. [1], to be more convincing, we decided to test this assumption experimentally. To do this, we fixed two arbitrary values $\mathbf{x}, \mathbf{x'} \in \{0,1\}^{128}$, and generated uniformly at random ℓ keys $\mathbf{k}_1, \ldots, \mathbf{k}_\ell$. Feeding $\mathbf{x}$ and $\mathbf{x'}$ into AES-128 with keys $\{\mathbf{k}_i\}_{i=1}^{\ell}$ we get a sample of pairs $[F_{\mathbf{k}_i}(\mathbf{x}), F_{\mathbf{k}_i}(\mathbf{x'})]_{i=1}^{\ell}$. Next, we test whether the distribution from which this sample was generated is ϵ-close to uniform

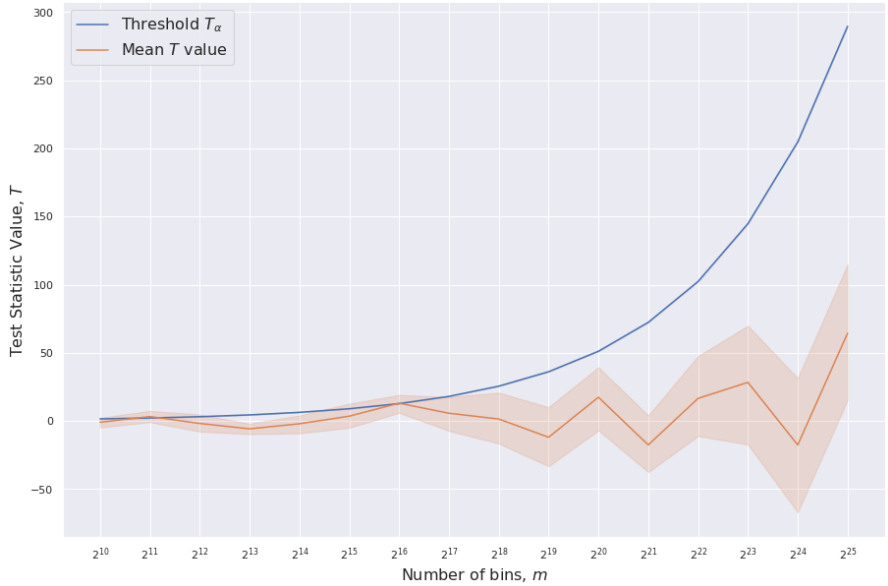

Figure 1: Results of testing closeness of AES outputs to pairwise independence. Shaded region indicates 90% confidence band across 10 runs of the test for each of the bin sizes.

distribution over distinct pairs of 128-bit strings. In total there are $2^{128} \cdot (2^{128} - 1)$ such pairs and treating each of them as a bin is not tractable. Therefore, we split them into bins so that the total number of bins allows for calculations on a regular desktop PC. Note that even after this procedure, if the bins are not too large, the sample size ℓ is usually still much less than the total number of bins m. And this means that classical tests based on the chi-square distribution in this case are not suitable. Therefore, we used the test proposed by Paninski [4], which is just suitable for the case $\ell \ll m$. Formally, let $p(j)$ be the true probability that a random vector $[F_{\mathbf{k}}(\mathbf{x}), F_{\mathbf{k}}(\mathbf{x}')]$ is in the j-th bin. Then to test the hypothesis

$$\mathrm{H}_0 \ : \ p(j) \equiv \frac{1}{m}, \qquad \forall j \in \{1, \dots, m\}$$

versus

$$\mathrm{H}_1 \ : \ \sum_{j=1}^{m} \left| p(j) - \frac{1}{m} \right| > \epsilon$$

we reject the null if

$$T \ := \ell \left(\frac{m-1}{m} \right)^{\ell - 1} - K_1 > T_\epsilon,$$

where T is the test statistic, K_1 is the number of bins into which just one sample has fallen, and T_ϵ is the critical value that depends on ϵ (we refer the reader to [4] for the details). In our experiments, we set $\epsilon = 0.01$, $\ell = m^{3/4}$, and vary m from 2^{10} to 2^{25} with an exponential step. For each m we perform 10 runs, i.e. we take a sample of size ℓ 10 times, and compute the values of test statistics for each run. The choice of $\mathbf{x}$ and $\mathbf{x}'$ is specified in the code attached[2]. The results

[2]https://bit.ly/3AM77nf

of evaluation are provided in Figure 1. The blue curve corresponds to the threshold value, and the orange one indicates the average of T accross 10 runs. The shaded band around the orange curve is the 90% confidence band. As we see, the statistical test of [4] fails to reject the null, especially when the number of bins grows, which supports Assumption 1.

4. Conclusion

Inspired by the recent result of Liu et al. [1] on statistical closeness of AES to pairwise independence under randomness of all round keys, we make a relevant assumption on *computational* closeness of AES to pairwise independence under randomness of just the initial key. Under this assumption we prove the resistance of AES against attacks based on machine learning algorithms that aim to recover AES key from pairs of ciphertexts. Our proof is elementary and uses only college-level probability. We argue that Assumption 1 is realistic and is a reasonable alternative to common cryptographic assumptions such as existence of a one-way function.

Acknowledgements

This work was supported by the Program of Targeted Funding "Economy of the Future" #0054/ПЦФ-НС-19.

References

[1] T. Liu, S. Tessaro, V. Vaikuntanathan, The t-wise independence of substitution-permutation networks, in: T. Malkin, C. Peikert (Eds.), Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV, volume 12828 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 454–483. URL: https://doi.org/10.1007/978-3-030-84259-8_16. doi:10.1007/978-3-030-84259-8_16.

[2] J. Daemen, V. Rijmen, The block cipher rijndael, in: J. Quisquater, B. Schneier (Eds.), Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings, volume 1820 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 277–284. URL: https://doi.org/10.1007/10721064_26. doi:10.1007/10721064_26.

[3] B. Tao, H. Wu, Improving the biclique cryptanalysis of AES, in: E. Foo, D. Stebila (Eds.), Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings, volume 9144 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 39–56. URL: https://doi.org/10.1007/978-3-319-19962-7_3. doi:10.1007/978-3-319-19962-7_3.

[4] L. Paninski, A coincidence-based test for uniformity given very sparsely sampled discrete data, IEEE Trans. Inf. Theory 54 (2008) 4750–4755. URL: https://doi.org/10.1109/TIT.2008.928987. doi:10.1109/TIT.2008.928987.